Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Telemática

2021

Trabajo Fin de Grado

Despliegue de OSM (Open Source MANO) sobre un entorno Openstack para la implementación de servicios NFV

Autor: Miguel Herrero Larena
Tutor: Isaías Martínez Yelmo

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Ingeniería Telemática

Trabajo Fin de Grado

Despliegue de OSM (Open Source MANO) sobre un entorno Openstack para la implementación de servicios NFV

Autor: Miguel Herrero Larena

Tutor/es: Isaías Martínez Yelmo

TRIBUNAL:

Presidente: Elisa Rojas Sánchez

Vocal 1º: José Manuel Arco Rodríguez

Vocal 2º: Isaías Martínez Yelmo

FECHA: 29/07/2021

Resumen

La industria de las telecomunicaciones ha venido evolucionando, adaptándose a las necesidades del mercado, uno de los siguientes pasos es la búsqueda de un modelo de arquitecturas más flexibles y adaptativas. En este proyecto planteamos una arquitectura compuesta por cuatro máquinas con el objetivo de desplegar servicios NFV , así como su despliegue y configuración automatizada la cual tiene como componentes principales Open Source Mano y Openstack y su objetivo principal no es otro que desplegar servicios NFV.

Open Source MANO por su parte, llevará a cabo la tarea de orquestación de dicha infraestructura. Será el encargado de ordenar las acciones relacionadas con los ciclos de vida de los servicios NFV (despliegue, modificación y eliminación). Para ello, hará uso de Openstack, que será el encargado de proporcionar la infraestructura para poder prestar servicios NFV.

Abstract

The telecommunications industry has been evolving, adapting to the needs of the market, one of the next steps is the search for a more flexible and adaptive architecture model. In this project we propose an architecture composed of four machines with the objective of deploying NFV services, as well as their automated deployment and configuration, which has as main components Open Source Mano and Openstack and its main objective is none other than to deploy NFV services.

Open Source MANO for its part, will carry out the task of orchestrating said infrastructure. It will be in charge of ordering the actions related to the life cycles of the NFV services (deployment, modification and elimination). To do this, it will make use of Openstack, which will be in charge of providing the infrastructure to be able to provide NFV services.

Índice de contenido

RESUMEN ..... **V**

ABSTRACT ..... **VII**

ÍNDICE DE ILUSTRACIONES ..... **XIII**

ÍNDICE DE TABLAS ..... **XV**

PALABRAS CLAVE ..... **XVII**

1. INTRODUCCIÓN **1**

2. DESCRIPCIÓN Y OBJETIVOS **5**

3. ESTADO DEL ARTE **7**

Network Function Virtualization *7*

Arquitectura NFV 7

Network Functions Virtualization Infraestructure (NFVI) 8

Virtualized Network Functions (VNFs) 8

Management and Orchestration (MANO) 8

Entidades NFV 9

OSM 9

OPENSTACK 10

Servicios existentes *11*

4. DISEÑO DE LA ARQUITECTURA A DESPLEGAR **15**

ARQUITECTURA 15

Servicios desplegados *16*

Nova 16

Neutron 16

Cinder 16

Keystone 17

Glance 17

Placement 17

Heat 17

Horizon 17
REQUISITOS HARDWARE Y SOFTWARE 17
ARQUITECTURA DE RED 20
5. DESPLIEGUE Y CONFIGURACIÓN DE LA ARQUITECTURA 21
OSM 21
Método de despliegue 21
Kubernetes 22
Docker swarm 22
Despliegue de OSM 23
Comprobación 23
Problemas encontrados y soluciones planteadas 25
Openstack 27
Instalación y configuración de KVM 27
Configuración del deployment host 28
Configuración de dependencias 28
Configuración de red 28
Clonar repositorios 28
Despliegue y configuración de los hosts 29
Configuración SSH. 32
Configuración de red. 32
Configuración del despliegue 32
Entorno de configuración 33
Credenciales 33
Ejecución de playbooks 33
6. INTEGRACIÓN DE AMBAS PLATAFORMAS 39
PROCESO DE INTEGRACIÓN 39
CONFIGURACIÓN Y ABASTECIMIENTO PARA EL TRABAJO CONJUNTO 42
Openstack 42
Creación de una red 43
Creación de un par de claves 45
Establecimiento de un flavour 46

Creación de una imagen 47

Open Source Mano *47*

7. COMPROBACIÓN DE RESULTADOS 49

8. CONCLUSIONES 59

9. BIBLIOGRAFÍA 61

ANEXO A. ARCHIVO NAT-NET.XML 65

ANEXO B. CONFIGURACIÓN DE RED DE LOS HOST 67

HOST DE DESPLIEGUE 67

HOST CONTROLLER 68

HOST COMPUTE: 72

HOST BLOCK STORAGE 74

HOST OSM 77

ANEXO C. ARCHIVO OPENSTACK_USER_CONFIG.YML 81

ANEXO D. ERROR SERVICIO PLACEMENT 89

ANEXO E. CONTENIDO ARCHIVO ADMIN-OPENRC.SH 95

ANEXO F. ERROR DE INTEGRACIÓN ENTRE OSM Y OPENSTACK 99

ANEXO G. REPOSITORIO GITHUB 101

ANEXO H. ARCHIVO DE CREDENCIALES 103

Índice de ilustraciones

Ilustración 1. Esquema Ansible. [13] 3
Ilustración 2. Esquema NFV. [16] 7
Ilustración 3. Esquema Openstack. [9] 10
Ilustración 4. Arquitectura Openstack. [17] 15
Ilustración 5. Snapshots de VirtualBox. 18
Ilustración 6. Arquitectura sistema entero. 19
Ilustración 7. Interfaz OSM, ... 24
Ilustración 8. Comporbación de la salud de OSM. 25
Ilustración 9. Ventana de diálogo de Ubuntu 30
Ilustración 10. Asignación de IP estática 31
Ilustración 11. Asignación de IP 31
Ilustración 12. Página de Login de Openstack. 35
Ilustración 13. Interfaz de Openstack. 36
Ilustración 14. Localiación del fichero RC. 39
Ilustración 15. Comprobación de los servicios desplegados de Openstack. 40
Ilustración 16. Localiación de la dirección URL del servicio Identity. 41
Ilustración 17. Visualización de la URL de Identity. 41
Ilustración 18. Comprobación del estado de la cuenta VIM en la API. 42
Ilustración 19. Comprobación del estado de la cuenta VIM en la terminal. 42
Ilustración 20. Creación de una red. 43
Ilustración 21. Creación de una subred. 43
Ilustración 22. Especificación de los detalles de la subred. 44
Ilustración 23. Creacion de una red en la terminal. 44
Ilustración 24. Creación de una subred en la terminal. 45
Ilustración 25. Creación de una par de claves. 45
Ilustración 26. Creación de Flavour 46

Ilustración 27. Creación de un Flavour mediante la terminal. 46
Ilustración 28. Creación de una imagen. 47
Ilustración 30. Creación de paquetes NS. 48
Ilustración 31. Creación de paquetes VNF. 48
Ilustración 32. Lanzamiento de una instancia en OSM. 49
Ilustración 33. Estado de la instancia lanzada en Openstack. 49
Ilustración 34. Comprobación del estado de los servicios Cinder. 52
Ilustración 35. Error de integración entre OSM y Openstack. 99

Índice de tablas

Tabla 1. Distribución de recursos. 19
Tabla 2. Redes de la arquitectura. 20
Tabla 3. Direcciones IP de las máquinas Openstack. 20
Tabla 4. Comparación OSM. 22
Tabla 5. Direcciones IP de las máquinas. 30
Tabla 6. Almacenamiento de las máquinas. 31

Palabras clave

OSM, openstack, NFV, ansible

1. Introducción

Tradicionalmente las redes de comunicaciones se han compuesto de una combinación entre software y hardware muy específicos en los que cualquier cambio ha supuesto una inversión importante de recursos (tiempo, dinero y personal). Otro detalle a tener en cuenta es el grado de especialización que los fabricantes de hardware han adoptado de cara a la construcción de los componentes de red, llevando a cabo cada uno de estos componentes una función muy específica y otorgando un nivel de rigidez realmente alto en la configuración y despliegue de las redes de comunicaciones.

Es aquí donde entra en juego la tecnología NFV(network function virtualization, virtualización de funciones de red) [1]. Esta tecnología viene a flexibilizar y agilizar la construcción y modificación de redes mediante la virtualización. Consiste en resolver mediante software las tareas que antes llevaban a cabo los equipos físicos de red. La arquitectura NFV se compone de tres capas: Infraestructura de virtualización de funciones de red (NFVi), las funciones de red virtual (VNF), que se compone de las aplicaciones que desempeñan las funciones de red, y la gestión, automatización y orquestación de redes (MANO) [2], que es la capa para la gestión y la orquestación de NFVi y varias NFV. NFV nos aporta beneficios como reducir el costo de las redes migrando de una infraestructura basada en elementos físicos a software y paralelamente, una reducción importe en el coste energético y en la utilización del espacio físico. No obstante, está tecnología requiere el uso de servidores de alto rendimiento con un alto grado de fiabilidad [3].

Por otro lado, NFV cuenta con otros beneficios a favor como el uso de un hardware mucho más genérico y no tan especializado como se venía usando. Siendo posible la ejecución simultanea de varias de estas aplicaciones localizadas en un mismo servidor. Pudiendo generarse altas en máquinas virtuales ante una necesidad o bajas de las mismas cuando ya no sea necesario seguir prestando un servicio. Otro aspecto importante para comentar es el pragmatismo que ofrece este hardware tan genérico en cuanto a reparaciones o dimensionado. En una realidad paralela en la que esta tecnología fuera mayoritaria en el mercado, estas máquinas serían muy comunes en el stock habitual, así como sus piezas principales, permitiendo una simplicidad sin precedentes a nivel técnico donde una sola máquina sustituiría la función de diferentes equipos específicos de red.

Podremos ver y comprobar las posibilidades de NFV, todo ello según los marcos de ETSI (European Telecommunications Standards Institute) y dentro de la capa MANO.

MANO es el responsable se la interacción entre todas las entidades dentro de la arquitectura NFV, comunicándose tanto con la NFVi como con las VNFs. El ETSI definió la arquitectura MANO para facilitar la transición de los equipos desde su desacople de los equipos físicos dedicados a las nuevas máquinas virtuales [4].

Open Source MANO [5] por su parte, es la implementación de MANO a partir de herramientas de código abierto, todo ello para la facilitar la implementación de una arquitectura NFV alineada con ETSI. Esta tecnología tiene como objetivo diseñar un ecosistema de proveedores de soluciones NFV que brinde sus servicios a un conjunto de usuarios de manera ágil y rentable, permitiendo la capitalización de la sinergia entre la estandarización y los enfoques de código abierto al acceder a un conjunto de colaboradores y desarrolladores más amplio y diverso de lo que normalmente sería posible.

Por otro lado, tenemos la tecnología Openstack [6]. Se trata de una plataforma de código abierto que se emplea para diseñar y gestionar nubes privadas y públicas.

Más allá de la funcionalidad estándar de la infraestructura como servicio, los componentes adicionales brindan orquestación, administración de fallos y administración de servicios entre otras funcionalidades para garantizar una alta disponibilidad de las aplicaciones de usuario [7].

Concretamente y centrándonos más en este proceso, detallaremos el despliegue de la arquitectura anteriormente mencionada y comentada, así como las dificultades y errores encontrados.

Dicho despliegue será automatizado con la herramienta de automatización y configuración Ansible [8].

Ansible es un motor de código abierto que permite un alto nivel de automatización para preparar y gestionar configuraciones en diferentes máquinas de forma simultánea. Esta automatización que nos ofrece permite multitud de opciones cómo automatizar tareas diarias, generar rutinas de seguridad o abastecimiento, ejecutar parches en diferentes sistemas, preparar infraestructuras o desplegar herramientas. [9]

Uno de los grandes beneficios de Ansible es la posibilidad de compartir estos procesos automatizados mediante archivos conocidos como *playbooks.* Gracias a ellos, siempre y cuando la herramienta esté instalada en los equipos, una empresa podrá externalizar

la configuración o el despliegue de una herramienta desde un equipo de despliegue al resto de equipos de la compañía, suponiendo un gran ahorro de tiempo como de dinero.

En nuestro caso emplearemos dicha herramienta para brindar la posibilidad a otros usuarios de poder desplegar nuestro entorno de una forma práctica y sencilla una vez se posean los principales archivos del despliegue, los playbooks.

Ilustración 1. Esquema Ansible. [10]

En esta memoria será expuesto el proceso completo de despliegue, partiendo desde una máquina vacía, pasando por el despliegue de los host de despliegue y finalizando con un estado del software completamente operativo. A lo largo de la exposición de las configuraciones y despliegues serán argumentados datos como los recursos dedicados, servicios desplegados y herramientas fundamentales para poder llevar a cabo el objetivo.

2. Descripción y objetivos

El proyecto que a continuación se presenta consiste en la instalación y automatización del despliegue de Open Source Mano siguiendo las pautas del TFM (Trabajo de fin de máster) “Data-Driven resource orchestration in sliced 5G Networks” de Adrián Gallego Sánchez [3], aprovechando todos sus beneficios al ser una tecnología de código abierto y todo el soporte que le ofrecen las grandes empresas del sector de las telecomunicaciones y todo ello sobre Openstack. Uno de los principales objetivos de este TFG es aprovechar al máximo las posibilidades de automatización de la herramienta Ansible para abstraer al máximo al usuario que vaya a proceder con este despliegue de las configuraciones de más bajo nivel necesarias para que todo funcione correctamente.

Open Source MANO será empleado para la dirección y orquestación. Recientemente, se ha vuelto mucho más ágil con su versión 10. OSM destaca por su pragmatismo de cara a la nube, lo cual lo hace idóneo para operar junto a Openstack, el cual posee una arquitectura realmente probada para las nubes públicas y privadas.

La implementación de OSM en la infraestructura NFV se ha vuelto muy práctica y sencilla, por otro lado, Openstack se caracteriza por la simplicidad para administrar infraestructuras virtualizadas que se basan en contenedores. Las organizaciones y entidades pueden obtener amplios beneficios de la integración de NFV MANO utilizando OSM y Openstack debido a una sencilla administración y un despliegue asequible.

Los objetivos principales del proyecto son:

• El estudio de la arquitectura NFV y las diferentes tecnologías de las que hace uso, teniendo en cuenta las diferentes alternativas que existen de cara a la misma y en función de una serie de requerimientos que nos impone la solución final deseada.

• El estudio de los componentes, entidades y roles dentro de la arquitectura NFV, así como el estudio de las configuraciones que más se adaptan a lo que buscamos.

• Diseño y planificación. Una vez llevado a cabo el estudio de todas las posibilidades de implementación y automatización de elegirá el software o tecnología que más convenga dadas las pautas.

• Automatización del despliegue de un entorno NFV mediante el uso de herramientas devops:

o Haciendo uso de un entorno de virtualización basado en Openstack.

o Desplegando sobre el entorno virtualizado la implementación de Open Source Mano.

o Para la implementación se usará VirtualBox con el objetivo de desplegar las máquinas virtuales, una para OSM y otras tres para openstack. A continuación, se hará uso de Ansible. La herramienta anteriormente mencionada será empleada para la automatización de Openstack para luego definir unas pruebas básicas de la arquitectura desplegada.

• Uso exclusivo de herramientas de Código Abierto (Open Source). Como se ya se ha explicado anteriormente, para la capa MANO se empleará Open Source Mano, por otro lado, el gestor VIM (gestor de recursos virtualizados) que se va a emplear es Openstack.

• Validación de la solución. Una vez finalizada la implementación y configurada la automatización, se comprobará que los servicios están activos y que existe comunicación entre Open Source Mano y Openstack, y que, por lo tanto, son capaces de trabajar de forma de conjunta.

• Documentación de los resultados obtenidos. Una vez se haya desarrollado y validado todo el proceso, se llevará a cabo una documentación de todos los pasos llevados a cabo y todas las decisiones que se han ido tomando hasta la obtención de la solución.

3. Estado del arte

En este apartado se llevará a cabo un análisis de las diferentes tecnologías y software usados en este despliegue y la forma en la que interactúan entre ellas. También se analizará el estado actual de cada herramienta usada detallando sus servicios y las diferentes opciones que ofrecen.

Network Function Virtualization

“Es un modelo de arquitectura de red que tiene como objetivo la virtualización de servicios de la misma. Permite la sustitución de dispositivos de hardware especializados y caros, como routers, switches o firewalls, por funciones de red basadas en software que se ejecutan como máquinas virtuales en servidores genéricos. Esto implica que todas las funciones realizadas por los nodos de la red deben estar definidas en un modelo virtual, más conocidas como VNFs.” [11].

A día de hoy la tecnología NFV sigue un patrón en cuanto a estructura. A continuación, la analizaremos, así como todas sus partes.

Arquitectura NFV

En la introducción se han presentado estos conceptos desde un punto de vista bastante amplio, ahora nos veremos un poco más en profundidad para entender la función que desempeña cada uno. En el siguiente esquema se puede apreciar la arquitectura elemental de un servicio NFV:

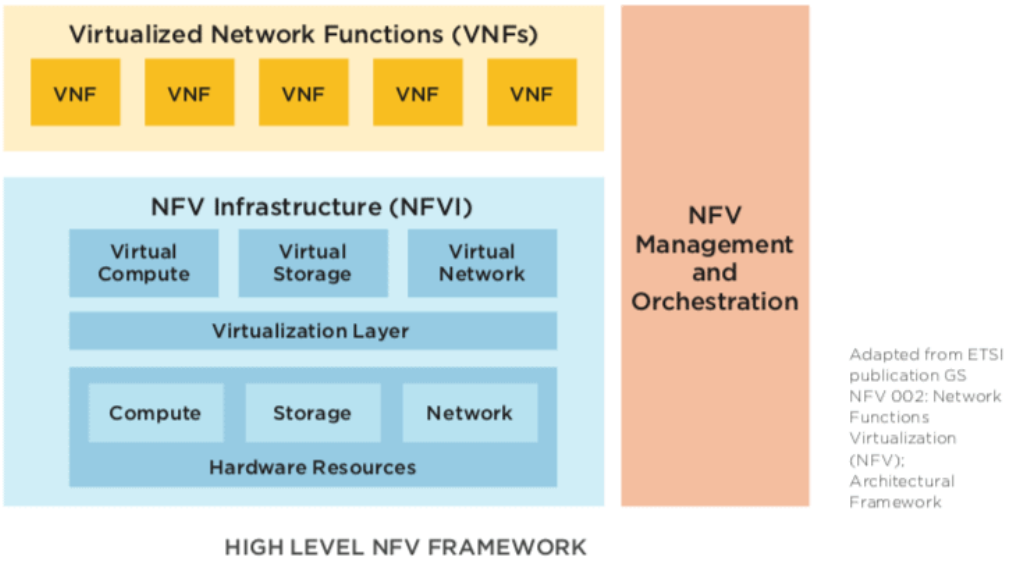

Ilustración 2. Esquema NFV. [12]

Tal y como podemos observar en la Ilustración 3, un servicio NFV se divide en tres partes, teniendo asignadas cada una de ellas una serie de funciones muy específicas y delimitadas:

Network Functions Virtualization Infraestructure (NFVI)

Este bloque cumple la función describir los componentes tanto software como hardware sobre los que se despliegan las redes virtuales.

Virtualized Network Functions (VNFs)

Las VNFs Aplican la configuración necesaria a las máquinas virtuales que ofrece NFVI para poder desarrollar las funciones visualizadas de red que ofrece la plataforma.

Management and Orchestration (MANO)

“MANO es un bloque separado dentro de la arquitectura NFV, que interactúa tanto con la NFVI como con las VNFs. Es el responsable del control de todas las entidades dentro de la arquitectura NFV. Por tanto, debe conocer el uso, el estado, las estadísticas y el resto de los parámetros asociados de todos los elementos implicados en la arquitectura.” [11].

Los bloques funcionales de MANO son los siguientes:

VNF Manager: su función principal es la gestión de las VNFs. se encarga de tareas como gestionar el ciclo de vida de las VNFs, gestión de fallos rendimiento configuración y seguridad y redimensionado de estas mismas.

NFV Orchestrator: El competidor principal de este componente es la gestión de los NS, es decir, la gestión del ciclo de vida de los NS, abastecer la conexión de extremo a extremo entre los diferentes VNF y atender las solicitudes de NFVI

Virtual Infraestructure Manager: Es el encargado de la gestión de los recursos de NFVI en cada dominio. En una arquitectura NFV se puede dar el caso de la coexistencia de varios VIM, haciéndose cargo cada uno de gestionar y abastecer su propio dominio NFVI. este bloque lleva a cabo funciones como la gestión del ciclo de vida de los recursos virtuales en su NFVI, gestionar un inventario de las VMs asociadas al uso de la infraestructura que está haciendo cada una, gestión de errores asociados al software, hardware y otros recursos virtuales.

También es importante comentar el concepto de *entidad NFV,* así como detallar las dos principales entidades:

Entidades NFV

Dentro de una arquitectura NFV existen dos principales entidades que serán gestionadas de una forma u otra en función de las necesidades.

Virtual Network Function (VNF): se trata de una infraestructura software cuya función es llevar a cabo la función de diferentes elementos de red como pueden ser un router o un switch. La creación de estos elementos viene dada por el uso de VNFDs (fragmentos de código que establecen los parámetros principales, también conocidos como descriptores).

Network Service (NS): es un conjunto de VNFs. su comportamiento viene definido por él tipo de VNFs de los que está compuesto. Su creación se lleva a cabo de la misma forma que en el caso de los VNFs, por medio de los NSDs, Que una vez más consisten en fragmentos de código que marcarán las características principales de nuestro NS,

OSM

Vamos a proceder a desarrollar una serie de conceptos que son importantes para entender el funcionamiento de Open Source MANO.

Es importante entender que Open Source MANO es el servicio de código abierto que lleva a cabo las tareas del bloque MANO de la arquitectura NFV.

El bloque MANO está formado por tres partes:

- VNF Manager (VNFM): Es el encargado de la gestión de las VNF.
- NFV Orchestrator (NFVO): Es el encargado de gestionar las NS.
- Virtual Infraestructure Manager (VIM): Es el encargado de gestionar los recursos NFVI en un dominio.

MANO trabaja con diferentes perfiles de funciones de red virtual estándar para que los usuarios puedan elegir entre los recursos de infraestructura de virtualización de funciones de red (NFVi) más adecuados para implementar su plataforma NFV [13]. Para que NFV MANO sea eficaz, debe integrarse con interfaces de programas de aplicación (API) en sistemas existentes para trabajar con tecnologías de múltiples proveedores en

múltiples dominios de red. MANO también debe interactuar con los sistemas de facturación y operaciones de los proveedores de telecomunicaciones [14].

Openstack

En este apartado veremos los servicios de los que está compuesto Openstack y su funcionalidad dentro de la infraestructura. Cada uno de estos servicios tiene una tarea muy concreta asignada, permitiendo un alto grado de organización.

Las herramientas que componen la plataforma gestionan los principales servicios de cloud computing. A continuación, tenemos una imagen bastante intuitiva que nos permite hacernos una idea más amplia de Openstack y su estructura básica.

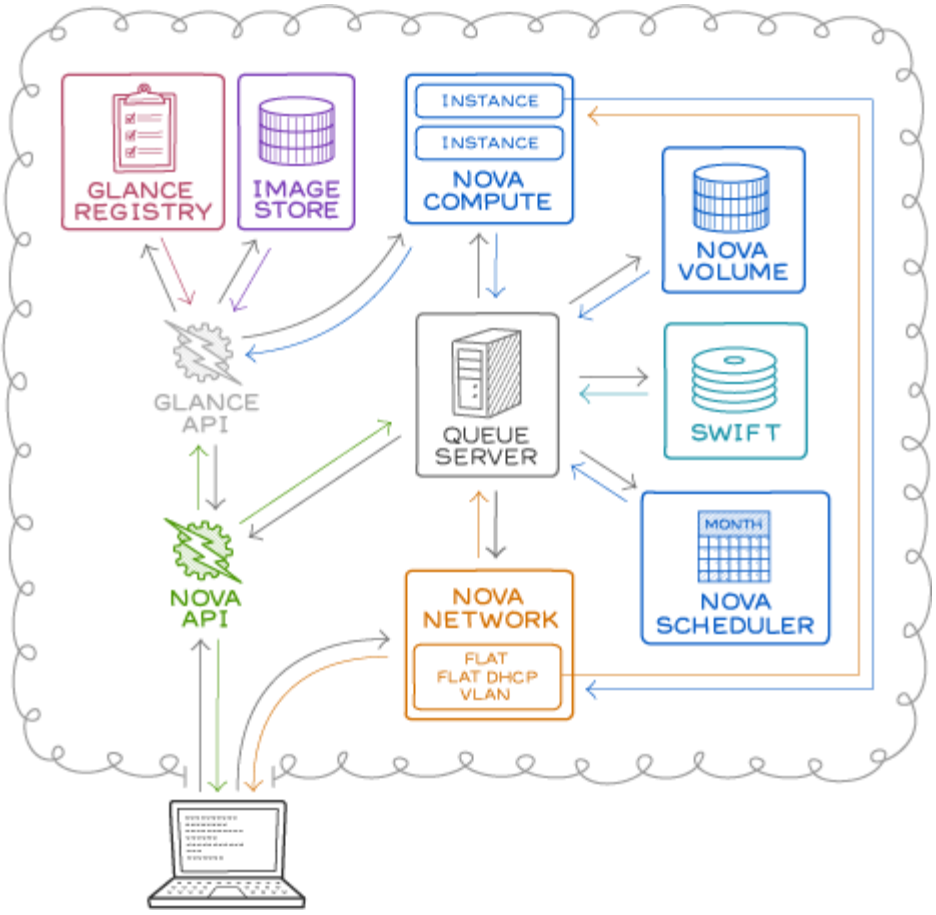

Ilustración 3. Esquema Openstack. [15]

Openstack posee un panel de control, que permite a los administradores gestionar el control al mismo tiempo que permite a sus usuarios aprovisionar recursos a través tanto de una interfaz web como desde la terminal de máquina principal del sistema, brindando

estas dos posibilidades de un amplio abanico de opciones y posibilidades para la plataforma.

Hoy en día, existen diversos métodos de instalación de Openstack en función del método empleado. Nosotros nos hemos decidido por el método basado en ansible debido a sus numerosas ventajas:

- Grado de automatización,
- Permite la opción de aprovisionar máquinas y servicios.
- El mantenimiento recae sobre la propia plataforma, la cual mantiene los playbooks actualizados y repara los bugs que se puedan detectar a lo largo de los años de mantenimiento.
- Nivel de abstracción en cuanto al proceso.

A continuación, veremos algunos de los servicios existentes en Openstack, así como sus funciones principales.

Servicios existentes

Es importante conocer los servicios más importantes de Openstack para poder hacernos una idea de sus dimensiones y posibilidades, por ello, enumeraremos y explicaremos brevemente los más usados y conocidos clasificándolos según el papel que desempeñan dentro del sistema:

• **Computación**

o Nova: permite implementar servicios por medio de peticiones de forma escalable y masiva a los recursos informáticos, incluidos los contenedores, las máquinas virtuales y los contenedores.

o Zun: proporciona una API Openstack para lanzar y administrar contenedores respaldados por diferentes tecnologías

• Ciclo de vida del hardware

o Ironic: servicio de aprovisionamiento que permite proporcionar acceso de forma escalable y masiva a los recursos informáticos, incluidos los contenedores y las máquinas virtuales.

o Cyborg: proporciona un marco de gestión de propósito general para aceleradores (incluidas GPU, FPGA, dispositivos basados en ASIC, etc.)

• **Almacenamiento**

o Swift: es un almacén de objetos con una alta disponibilidad, distribuido y eventualmente coherente. Los usuarios pueden usar Swift para almacenar gran cantidad de datos de manera eficiente, segura y económica.

o Cinder: es un servicio de almacenamiento organizado en bloques. Virtualiza la gestión de dispositivos de almacenamiento y proporciona a los usuarios finales una API para solicitar y consumir esos recursos sin la necesidad de saber dónde se encuentra realmente implementado su almacenamiento o en qué tipo de dispositivo.

• **Redes**

o Neutron: es un proyecto de redes SDN centrado en la entrega de redes como servicio (NaaS) en entornos informáticos virtuales.

o Octavia: Octavia es una solución de equilibrio de carga a escala de operador de código abierto diseñada para funcionar con OpenStack.

• Servicios compartidos

o Keystone: es un servicio que proporciona autenticación de cliente, descubrimiento de servicios y autorización distribuida mediante la implementación de la API de identidad.

o Placement: proporciona una API HTTP para realizar un seguimiento de los inventarios y usos de los recursos en la nube para ayudar a otros servicios a gestionar y asignar sus recursos de forma eficaz.

o Glance: permite descubrir, registrar y recuperar imágenes de máquinas virtuales. También tiene una API RESTful que permite la consulta de metadatos de imágenes de VM, así como la recuperación de la imagen real.

• Orquestación

o Heat: organiza los recursos de infraestructura para una aplicación en la nube basada en plantillas en forma de archivos de texto que se pueden tratar como código.

• Aprovisionamiento de cargas de trabajo

o Magnum: hace que los motores de orquestación de contenedores como Docker Swarm, Kubernetes y Apache Mesos estén disponibles como recursos de primera clase.

• Ciclo de vida de la aplicación

o Masakari: proporciona el servicio de alta disponibilidad de instancias para nubes recuperando automáticamente las instancias fallidas.

• Interfaz web

o Horizon: es la implementación del panel de control de OpenStack, que es extensible y proporciona una interfaz de usuario basada en web para los servicios.

4. Diseño de la arquitectura a desplegar

En este apartado detallaremos la arquitectura que se ha propuesto para el despliegue, explicando el número de máquinas que serán necesarias, los recursos con los que deberemos desplegar dichas máquinas y por supuesto, los servicios que albergará cada máquina.

Arquitectura

La arquitectura planteada para la infraestructura de Openstack responderá al siguiente esquema. Tal y como podemos observar en la imagen, se compondrá de tres máquinas con tres propósitos muy distintos. Por un lado, la máquina controller (Infrastructure Control Plane Host) se encargará de proporcionar el servicio de la API web, almacenar las imágenes de los elementos de red virtuales y de los servicios de identidad. Por otro lado, la maquina Compute (compute Host) alberga los servicios para poder gestionar los elementos virtuales de red, como su ciclo de vida (creación, destrucción o modificación) y los datos asociados a los mismos. Finalmente, el nodo de almacenamiento (Block Storage Host) proporciona sistemas de almacenamiento en bloque a las instancias, pudiendo soportar diferentes métodos de aprovisionamiento o método de consumo del almacenamiento.

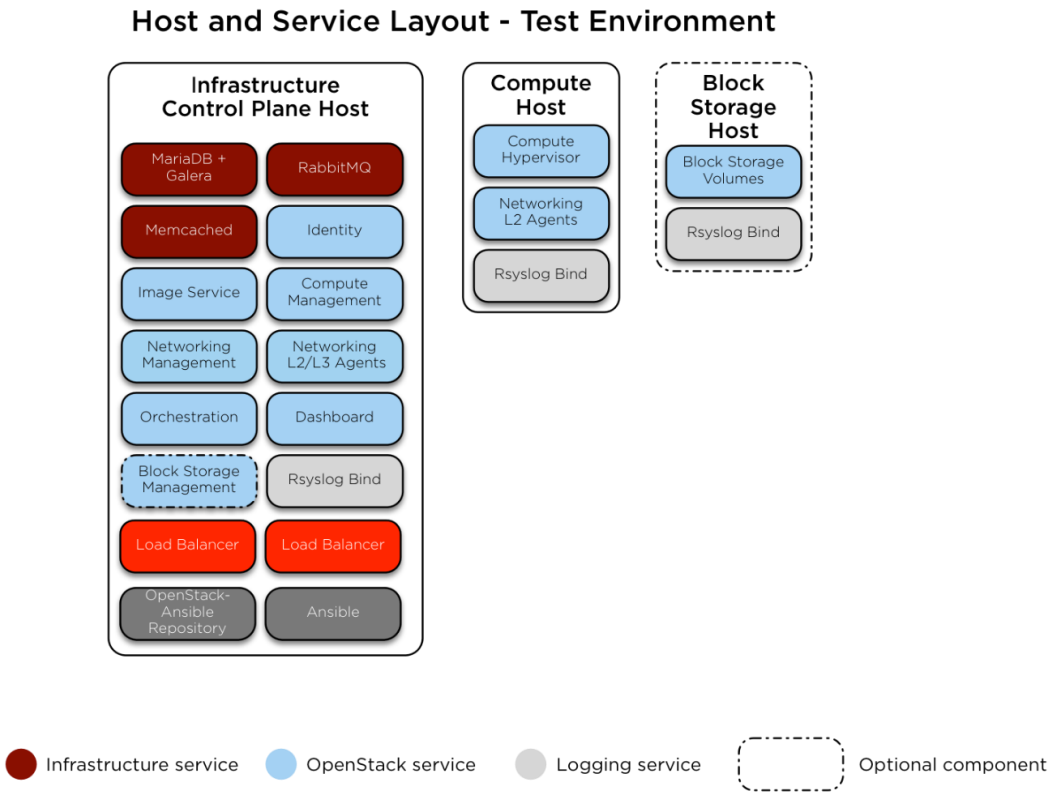

Ilustración 4. Arquitectura Openstack. [16]

Las tres máquinas que albergarán los servicios de Openstack coexistirán con una cuarta máquina que servirá para contener a Open Source MANO. Este último servicio será el encargado de ordenar el aprovisionamiento de los elementos de red en función de las necesidades del administrador.

Estas cuatro máquinas se encontrarán situadas en una quinta máquina virtual a la cual nos referiremos como “servidor”. La razón de ser de esta última máquina es para poder hacer el despliegue en un entorno Linux y tal y como se explicará en el siguiente apartado, para poder llevar un sistema de versiones de la máquina que contiene todo el despliegue, ya que el ordenador donde se ha llevado a cabo el despliegue es un ordenador Windows.

Dentro de Openstack se han desplegado los siguientes servicios en función de las tareas que llevan a cabo.

Servicios desplegados

Nova

Es el principal elemento de nuestro sistema, gestionará toda la parte de Cloud Computing. Administra los recursos y soporta diferentes tecnologías de virtualización (KVM, LXC o Hyper-V)

Neutron

Se encarga de la parte del Networking, gestionando así las redes desplegadas otorgando la posibilidad a los usuarios de crear, editar o eliminar redes tanto vía API como por terminal. También ofrece servicios como el control de tráfico, conexión de servidores y dispositivos en la UI.

Por otro lado, tiene asignado la seguridad de las redes, dando la posibilidad de desplegar cortafuegos, VPN (redes privadas), detección de intrusos o balanceos de carga entre otros servicios.

Cinder

Este servicio nos proporciona el almacenamiento necesario basado en bloques. Permite a los usuarios manipular dichos bloques para diseñar sus propios sistemas de almacenamiento. Es capaz de emplear el propio almacenamiento del sistema operativo, así como otros medios o tecnologías.

Keystone

Su tarea principal es brindar un servicio de identidad y autenticación al sistema, ofreciendo a día de hoy la posibilidad de emplear diferentes medios de autenticación.

Glance

Nos aporta un servicio de gestión de imágenes de disco, empleado fundamentalmente para copias de seguridad, servicios de entrega y registro, así como plantillas para instanciar.

Placement

Este servicio provee al sistema de una API vía HTTP que será empleada para llevar el seguimiento de inventarios, recursos y diferentes usos dentro del sistema como la memoria, la RAM o la red.

Heat

Nos da la posibilidad de orquestar múltiples aplicaciones situadas en la nube por medio de plantillas.

Horizon

El cometido de este servicio no es otro que nutrir al sistema de una interfaz gráfica que permitirá a usuarios y administradores gestionar dicho sistema. Ofrece características muy interesantes como la automatización y provisión de recursos basados en el cloud, así como servicios de monitorización o facturación.

Requisitos Hardware y Software

El hardware empleado ha sido un ordenador de mesa con las siguientes características:

- Procesador: AMD Ryzen 5 3600X (8 núcleos 16 hilos)
- Memoria RAM: 32 GB
- Disco: 1 TB HDD
- Tarjeta gráfica: Dedicada.

Todo el despliegue ha sido llevado a cabo en una máquina virtual creada en VirtualBox con un sistema Linux. Esta decisión fue tomada debido al pragmatismo que ofrece dicho

sistema de virtualización de cara a las capturas del estado de las máquinas en un momento determinado (snapshots), así como por las exportaciones y almacenamiento de máquinas completas ya configuradas o en un estado avanzado. Este concepto ha sido fundamental, ya que nos ha permitido desarrollar una plataforma de pruebas formada por una amplia biblioteca de capturas de diferentes estados de la máquina, permitiéndonos movernos en el tiempo retrocediendo y avanzando en el proceso de despliegue y configuración en función de las necesidades de modificación o almacenamiento.

También ha resultado muy práctico para la documentación de todo el proceso, permitiendo volver a ejecutar instrucciones para poder capturar la salida de la terminal o recopilar parámetros para publicar en esta memoria.

En esta imagen se muestran todos los *snapshots* realizados durante el proyecto:

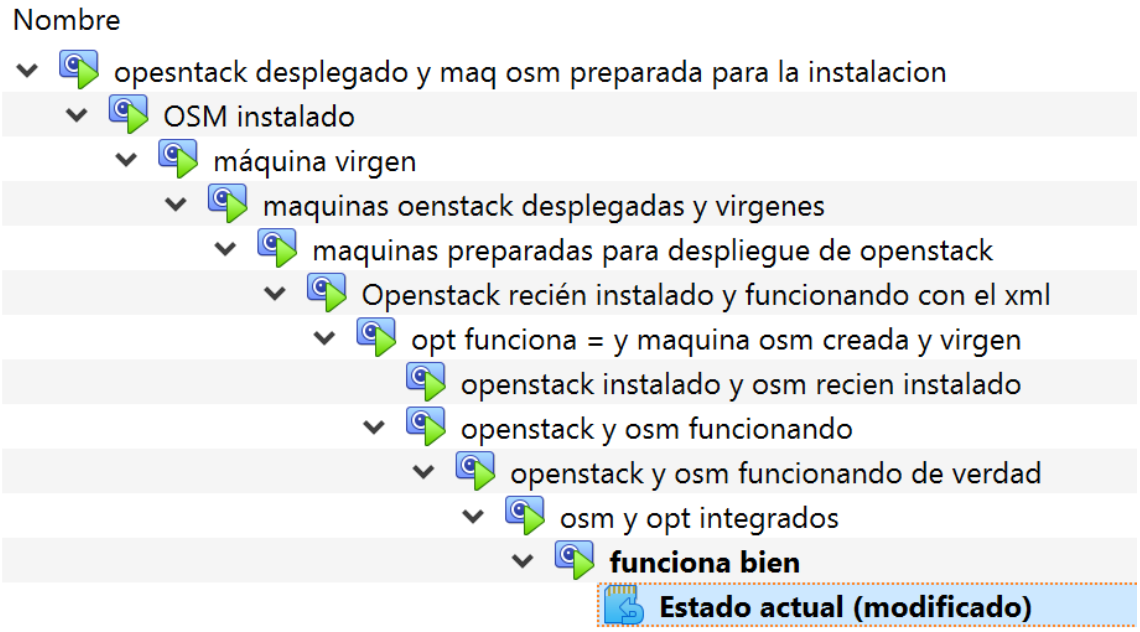

Ilustración 5. Snapshots de VirtualBox.

Todas las máquinas, tanto el servidor (host de despliegue) que albergará el resto de las máquinas como las máquinas donde serán desplegados los servicios, serán aprovisionadas con Ubuntu 20.04.2 LTS, siendo la versión Desktop en el caso del servidor.

Por otro lado, a nivel de hardware encontraremos la siguiente distribución:

Maquina	CPUs	RAM	Disco	Interfaces de red
Servidor	8	28 GB	800 GB	2
Controller	2	8 GB	150 GB	2
Compute	2	8GB	150 GB	3
Block Storage	1	4 GB	100 GB	
OSM	2	6 GB	200 GB	2

Tabla 1. Distribución de recursos.

Quedando una arquitectura con la forma que se puede ver en ilustración 6, observando 3 niveles de virtualización:

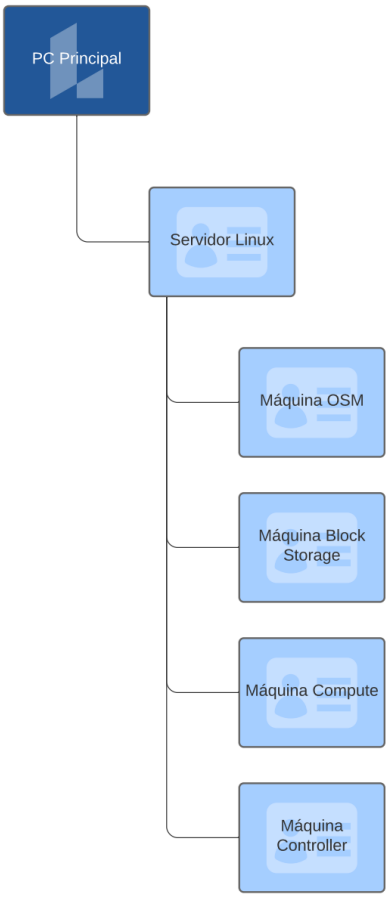

Ilustración 6. Arquitectura sistema entero.

Arquitectura de red

Tal y como nos indica la documentación oficial, hay que desplegar una serie de interfaces en cada máquina en función del papel que desempeñe en la arquitectura anteriormente planteada.

La red va a estar dividida en tres principales interfaces, tal y como podemos ver en la siguiente tabla:

Network	CIDR	VLAN
Management Network	172.29.236.0/22	10
Tunnel (VXLAN) Network	172.29.240.0/22	30
Storage Network	172.29.244.0/22	20

Tabla 2. Redes de la arquitectura.

Y la distribución de direcciones IP para el despliegue es la siguiente:

Host name	Management IP	Tunnel (VxLAN) IP	Storage IP
Controller	172.29.236.11	172.29.240.11	
Compute	172.29.236.12	172.29.240.12	172.29.244.12
Block Storage	172.29.236.13		172.29.244.13

Tabla 3. Direcciones IP de las máquinas Openstack.

Esta configuración será explicada individualmente para cada máquina en la sección de red específica de cada máquina dentro del apartado del despliegue. También se verá ampliada en el Anexo B.

5. Despliegue y configuración de la arquitectura

En este apartado veremos el despliegue de OSM y Openstack, haciendo uso de la herramienta Ansible. Ansible no requiere de más configuración que su propia instalación en el sistema y garantizar el alcance de los hosts donde se va a llevar a cabo el despliegue vía SSH.

OSM

Tal y como observaremos a continuación, vamos a ver las posibilidades que ofrece OSM para su despliegue e instalación y seguido de ello, será detallado todo el proceso siguiendo las directrices asumidas.

Método de despliegue

Open Source MANO ofrece dos principales métodos de despliegue. Ambos métodos han sido probados, evaluando el resultado obtenido, así como otros factores como los requisitos o el grado de complejidad del despliegue.

- **Vagrant:** donde mediante virtualbox o KVM se desplegaba una imagen de vagrant con todas las funciones necesarias para el correcto funcionamiento de OSM.
- **Script:** OSM como entidad ofrece por otro lado un script de instalación y configuración de las herramientas necesarias, así como de los servicios OSM, la cual debe ser instalado en una máquina Linux (preferiblemente Ubuntu 18.04).
-

A continuación, se muestra una tabla en la que se compara las características de cada método:

	Vagrant	**Script**
Versiones de OSM.	4, 5 y 6	4, 5 ,6 ,7 ,8 y 9
Nivel de complejidad del método.	9/10	4/10

Nivel de configuración requerida.	8/10	4/10
Requisitos software requeridos	8 GB de RAM y 100 GB de almacenamiento.	6 GB de RAM y 60 GB de almacenamiento.
Resultado del intento llevado a cabo.	Fallido. Errores en la configuración de red con fallos en la mayoría de los servicios a la hora de desplegarlos. El motivo principal lo asocio al nivel de encapsulamiento (doble virtualización) necesario y la complejidad de los servicios que se llevan a cabo en el interior de dicho encapsulamiento.	Exitoso. El software despliega sin problemas después de ajustar algunos parámetros y solventar algunos problemas.

Tabla 4. Comparación OSM.

Por todos estos motivos es por lo que finalmente se decidió llevar a cabo el despliegue con el Script y no con la imagen de vagrant, siendo determinantes los errores y las complicaciones con este segundo método.

OSM ofrece fundamentalmente dos métodos de instalación:

Kubernetes

Es el método por defecto que se establecerá automáticamente si ejecutamos el script siguiendo los pasos establecidos en la documentación oficial. Cualquier método de instalación que ejecutemos nos permitirá establecer una opción en la cual podremos redirigir todas las entradas y salidas al sistema hacia un archivo que quedará guardado en forma de log.

Docker swarm

El instalador desplegará y configurará los paquetes y herramientas requeridas para poder correr un nodo singular docker swarm y poder desplegar los diferentes objetos y servicios en él.

El comando asociado a esta opción de instalación es el siguiente:

```
./install_osm.sh -c swarm
```

En nuestro caso, el método de instalación elegido ha sido el basado en kubernetes debido a su menor impacto en el hardware y su mayor simplicidad y pragmatismo.

A continuación, será detallado el proceso de instalación, despliegue y configuración de OSM especificando decisiones tomadas, problemas encontrados y los resultados obtenidos

Despliegue de OSM

OSM ofrece en su documentación oficial todos los pasos para el despliegue. [5]

En primer lugar, descargamos el script necesario para la instalación:

```
# wgethttps://osm-download.etsi.org/ftp/osm-9.0-
nine/install_osm.sh
```

En segundo lugar, le otorgamos permisos de ejecución:

```
# chmod +x install_osm.sh
```

Finalmente, ejecutamos el script:

```
# ./install_osm.sh
```

Comprobación

Una vez finalizada la instalación esperaremos un tiempo prudencial para evaluar el resultado. Llevaremos a cabo el proceso de comprobación que se nos indicia en la documentación oficial para cerciorarnos de que todo ha salido bien [17] .Transcurridos 10-15 minutos podremos acceder a la web en nuestro navegador introduciendo la dirección IP asociada, en nuestro caso 10.0.2.15. A continuación, aparecerá una página de Login en la cual introduciremos el usuario y la contraseña por defecto, admin y admin. Llegados a este punto, una vez nos loguemos se nos redirigirá a una interfaz web:

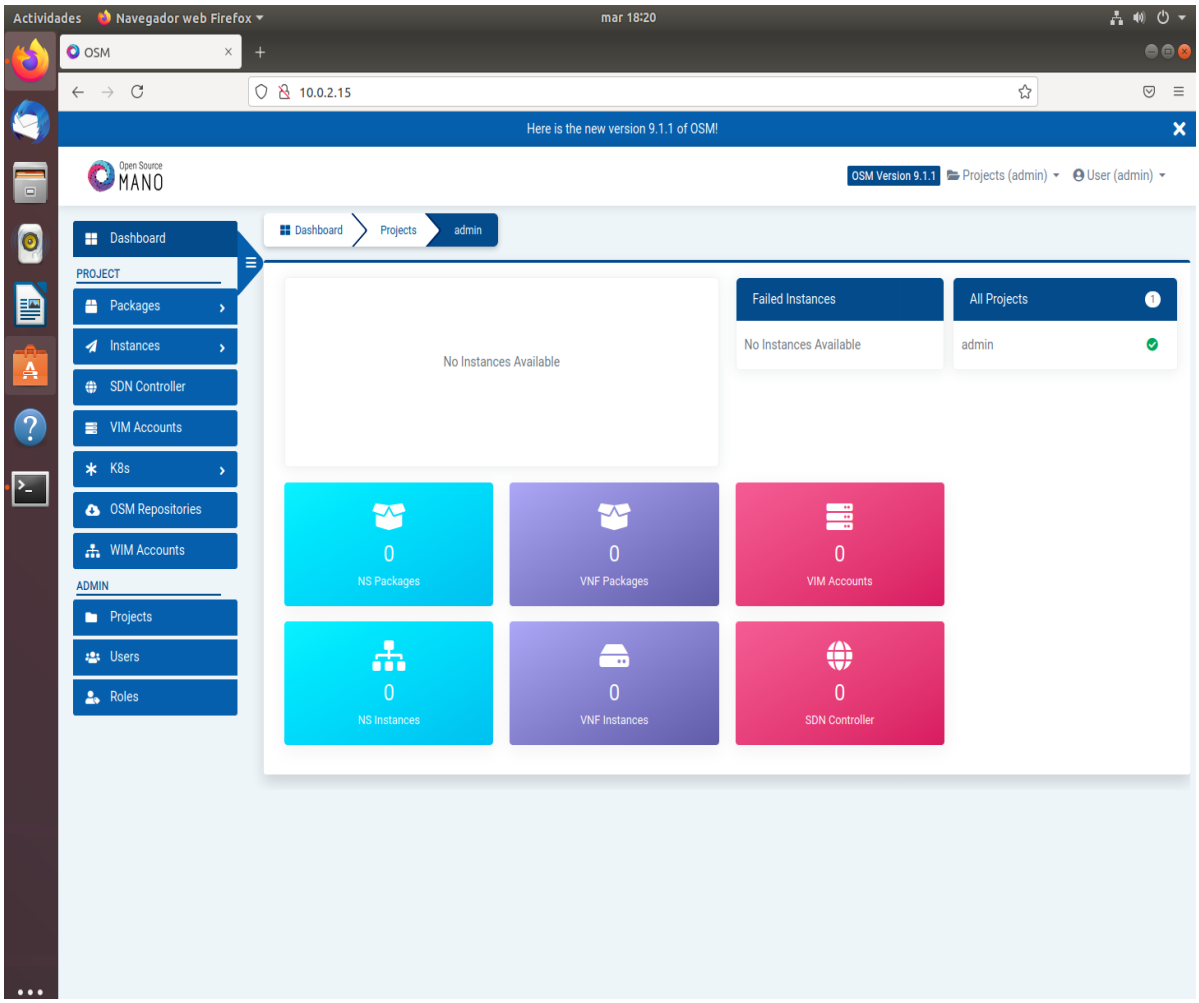

Ilustración 7. Interfaz OSM,

Por otro lado, desde la terminal de la máquina, introduciendo el siguiente comando:

```
# kubectl get all -n osm
```

Podremos comprobar el estado de todos los contenedores, así como las IPs asociadas. Para que todo esté correcto, la salida obtenida tendría que ser la siguiente:

```
root@osm:/home/osm# kubectl get all -n osm
NAME                                    READY   STATUS    RESTARTS   AGE
pod/grafana-755979fb8c-h82hc            2/2     Running   6          6d7h
pod/kafka-0                             1/1     Running   3          6d7h
pod/keystone-786dbb778d-8ckfz           1/1     Running   3          6d7h
pod/lcm-69d8d4fb5c-sbnc5                1/1     Running   8          6d7h
pod/modeloperator-6db944c5bb-zrpl7      1/1     Running   3          6d7h
pod/mon-7c966fccfb-5khff                1/1     Running   3          6d7h
pod/mongodb-k8s-0                       2/2     Running   6          6d7h
pod/mongodb-k8s-operator-0              1/1     Running   3          6d7h
pod/mysql-0                             1/1     Running   3          6d7h
pod/nbi-6c8b6dffb4-r6cr8                1/1     Running   3          6d7h
pod/ng-ui-6d55c58954-c59lc              1/1     Running   14         6d7h
pod/pol-66bff9db8c-qlq8t                1/1     Running   3          6d7h
pod/prometheus-0                        1/1     Running   3          6d7h
pod/ro-5d958cc7f9-tbc46                 1/1     Running   3          6d7h
pod/zookeeper-0                         1/1     Running   3          6d7h


NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/grafana                   NodePort    10.108.145.219   <none>        3000:3000/TCP    6d7h
service/kafka                     ClusterIP   None             <none>        9092/TCP         6d7h
service/keystone                  ClusterIP   None             <none>        5000/TCP         6d7h
service/modeloperator             ClusterIP   10.108.67.227    <none>        17071/TCP        6d7h
service/mon                       ClusterIP   None             <none>        8662/TCP         6d7h
service/mongodb-k8s               ClusterIP   10.103.220.224   <none>        27017/TCP        6d7h
service/mongodb-k8s-endpoints     ClusterIP   None             <none>        <none>           6d7h
service/mongodb-k8s-operator      ClusterIP   10.110.59.11     <none>        30666/TCP        6d7h
service/mysql                     ClusterIP   None             <none>        3306/TCP         6d7h
service/nbi                       NodePort    10.103.190.213   <none>        9999:9999/TCP    6d7h
service/ng-ui                     NodePort    10.101.112.185   <none>        80:80/TCP        6d7h
service/prometheus                NodePort    10.105.234.168   <none>        9090:9091/TCP    6d7h
service/ro                        ClusterIP   None             <none>        9090/TCP         6d7h
service/zookeeper                 ClusterIP   None             <none>        2181/TCP         6d7h


NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana           1/1     1            1           6d7h
deployment.apps/keystone          1/1     1            1           6d7h
deployment.apps/lcm               1/1     1            1           6d7h
deployment.apps/modeloperator     1/1     1            1           6d7h
deployment.apps/mon               1/1     1            1           6d7h
deployment.apps/nbi               1/1     1            1           6d7h
deployment.apps/ng-ui             1/1     1            1           6d7h
deployment.apps/pol               1/1     1            1           6d7h
deployment.apps/ro                1/1     1            1           6d7h

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/grafana-755979fb8c           1         1         1       6d7h
replicaset.apps/keystone-786dbb778d          1         1         1       6d7h
replicaset.apps/lcm-69d8d4fb5c               1         1         1       6d7h
replicaset.apps/modeloperator-6db944c5bb     1         1         1       6d7h
replicaset.apps/mon-7c966fccfb               1         1         1       6d7h
replicaset.apps/nbi-6c8b6dffb4               1         1         1       6d7h
replicaset.apps/ng-ui-6d55c58954             1         1         1       6d7h
replicaset.apps/pol-66bff9db8c               1         1         1       6d7h
replicaset.apps/ro-5d958cc7f9                1         1         1       6d7h

NAME                                    READY   AGE
statefulset.apps/kafka                  1/1     6d7h
statefulset.apps/mongodb-k8s            1/1     6d7h
statefulset.apps/mongodb-k8s-operator   1/1     6d7h
statefulset.apps/mysql                  1/1     6d7h
statefulset.apps/prometheus             1/1     6d7h
statefulset.apps/zookeeper              1/1     6d7h
```

Ilustración 8. Comporbación de la salud de OSM.

Problemas encontrados y soluciones planteadas

Problema 1: Al ejecutar el script de instalación anteriormente mencionado, aparece el siguiente mensaje en pantalla con la consecuente parada con de la ejecución:

```
An error occurred during the signature verification. The
repository is not updated and the previous index files will be
used. GPG error: https://packages.cloud.google.com/apt cloud-sdk
InRelease: The following signatures couldn't be verified because
the public key is not Disponible: NO_PUBKEY FEEA9169307EA071
NO_PUBKEY 8B57C5C2836F4BEB
```

Esto sucede porque no están actualizadas las claves públicas.

Solución Problema 1: para poder continuar con la instalación, tendremos que incluir las claves públicas que nos muestren por pantalla de la siguiente forma [18]:

Sea la salida por pantalla "`NO_PUBKEY 8B57C5C2836F4BEB`"

Ejecutaremos:

```
# sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
8B57C5C2836F4BEB
```

Problemas 2: una vez instalado OSM, tendremos que definir el archivo de configuración. Por alguna razón, durante la instalación omite esta tarea y por lo tanto, en el momento que reiniciemos los servicios o la máquina obtendremos esta salida al intentar mostrar el estado de los contenedores con el comando anteriormente comentado:

```
# kubectl get all -n osm

The connection to the server localhost:8080 was refused - did
you specify the right host or port?
```

La solución en este caso pasa por crear el directorio donde kubectl buscará dicho archivo, los copiamos y finalmente le otorgamos los permisos [19]:

```
# mkdir -p $HOME/.kube

# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Openstack

En este apartado veremos el proceso completo de despliegue de Openstack partiendo de una máquina Ubuntu. El proceso constará de una serie de pasos donde desplegaremos y configuraremos 4 máquinas virtuales, la máquina principal (levantada con VirtualBox) llevará el papel de host principal el cual albergará al resto de máquinas virtuales (levantadas por medio de KVM), en las cuales se desplegarán los servicios indicados por medio de la herramienta Ansible.

Instalación y configuración de KVM

La primera tarea que debemos llevar a cabo es la instalación de KVM y en mi caso, la activación de la *doble virtualización*. Esta opción de *doble virtualización* debe ser modificada en la interfaz de VirtualBox. En el caso de no tener esa casilla activada (sucede en algunos casos) en esta web [20] queda perfectamente explicado el proceso para activarlo a través del cmd de Windows.

En este caso, KVM será usado como hypervisor/VIM, por lo que tendremos que configurar un pool de almacenamiento en el que serán guardadas las imágenes ISO. En nuestro caso, todo el almacenamiento es local, por lo que crearemos un nuevo *storage pool* para guardar el ISO de instalación del sistema operativo que usaremos. [21]

```
# mkdir /var/lib/libvirt/isos

# virsh pool-define-as kvmpool --type dir --target
/var/lib/libvirt/isos

# virsh pool-autostart kvmpool

# virsh pool-start kvmpool
```

Por otro lado, configuraremos la red dentro de KVM para poder proporcionar conexión a nuestras máquinas:

```
# virsh net-define nat-net.xml

# virsh net-start NAT_mgmt

# virsh net-autostart NAT_mgmt
```

Configuración del deployment host

Es muy importante llevar a cabo una configuración y puesta a punto solida de cara a la máquina que llevará a cabo la función de “servidor”, en caso de realizar una mala configuración podríamos tener errores o fallos en un futuro difíciles de rastrear y relacionar con la causa.

Configuración de dependencias

En primer lugar, nos encargaremos de instalar los paquetes software necesarios, tal y como se nos indica en la documentación oficial:

```
# apt-get update

# apt-get dist-upgrade

# apt-get install aptitude build-essential git ntp ntpdate
openssh-server python-dev sudo
```

Configuración de red

En cuanto a la configuración de red, en principio solo tendremos que añadir la interfaz asociada a la red 172.29.236.0/22. Lo haremos por medio del archivo /etc/network/interfaces.

Clonar repositorios

Llegado el momento y previamente teniendo instalada y configurada la herramienta GitHub, clonaremos los repositorios donde se encuentran los archivos de configuración, así como los playbooks. Desplazándonos al directorio /opt/openstack-ansible (lo crearemos si no lo tenemos en la ruta indicada):

```
# git clone -b 17.1.12
https://git.openstack.org/openstack/openstack-ansible
/opt/openstack-ansible
```

Seguido de ello, ejecutamos el script Bootstrap:

```
# scripts/bootstrap-ansible.sh
```

Despliegue y configuración de los hosts

Una vez desplegado y configurado el servidor, llega el momento de levantar y aprovisionar las maquinas que albergarán los servicios que harán posible el funcionamiento del entorno. Tal y como hemos explicado en el apartado donde se definía la estructura y las características de cada máquina, comenzaremos por los comandos de despliegue de cada una de las máquinas.

```
# sudo virt-install --name controller --memory 8192 --vcpus 2 --
cdrom ubuntu-20.04.2-live-server-amd64.iso --disk
pool=kvmpool,size=150 --boot cdrom,fd,hd,network --network
bridge=virbr_vm --virt-type kvm --graphics
vnc,port=5901,listen=0.0.0.0 --os-type linux --os-variant
ubuntu20.04
```

```
# sudo virt-install --name compute --memory 8192 --vcpus 2 --
cdrom ubuntu-20.04.2-live-server-amd64.iso --disk
pool=kvmpool,size=150 --boot cdrom,fd,hd,network --network
bridge=virbr_vm --virt-type kvm --graphics
vnc,port=5902,listen=0.0.0.0 --os-type linux --os-variant
ubuntu20.04
```

```
# sudo virt-install --name block --memory 4096 --vcpus 1 --cdrom
ubuntu-20.04.2-live-server-amd64.iso --disk
pool=kvmpool,size=100 --boot cdrom,fd,hd,network --network
bridge=virbr_vm --virt-type kvm --graphics
vnc,port=5903,listen=0.0.0.0 --os-type linux --os-variant
ubuntu20.04
```

```
# sudo virt-install --name OSM --memory 6144 --vcpus 2 --cdrom
ubuntu-20.04.2-live-server-amd64.iso --disk
pool=kvmpool,size=200 --boot cdrom,fd,hd,network --network
bridge=virbr_vm --virt-type kvm --graphics
```

```
vnc,port=5904,listen=0.0.0.0 --os-type linux --os-variant
ubuntu20.04
```

Tras ejecutar cada uno de estos comandos nos aparecerá una ventana de dialogo de Ubuntu en la que debemos configurar una serie de parámetros como la IP estática, el almacenamiento o las credenciales.

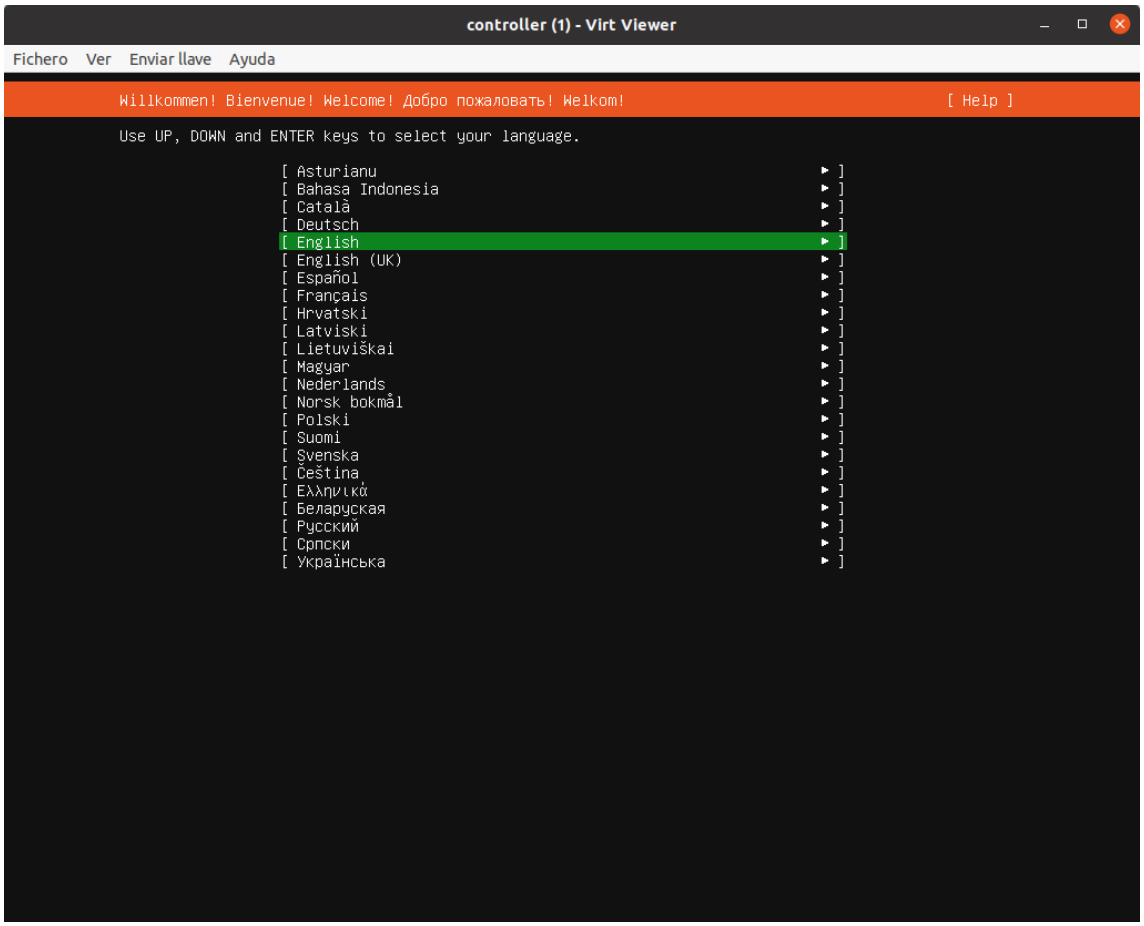

Ilustración 9. Ventana de diálogo de Ubuntu

Procedemos a configurar las direcciones táticas de red siguiendo la siguiente tabla:

Host name	Management IP
Controller	10.0.0.2
Compute	10.0.0.3
Block Storage	10.0.0.4
OSM	10.0.0.5

Tabla 5. Direcciones IP de las máquinas.

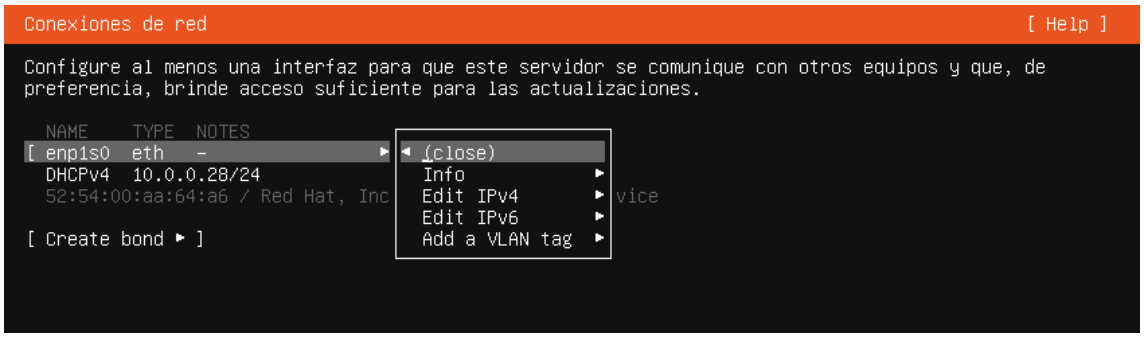

Ilustración 10. Asignación de IP estática

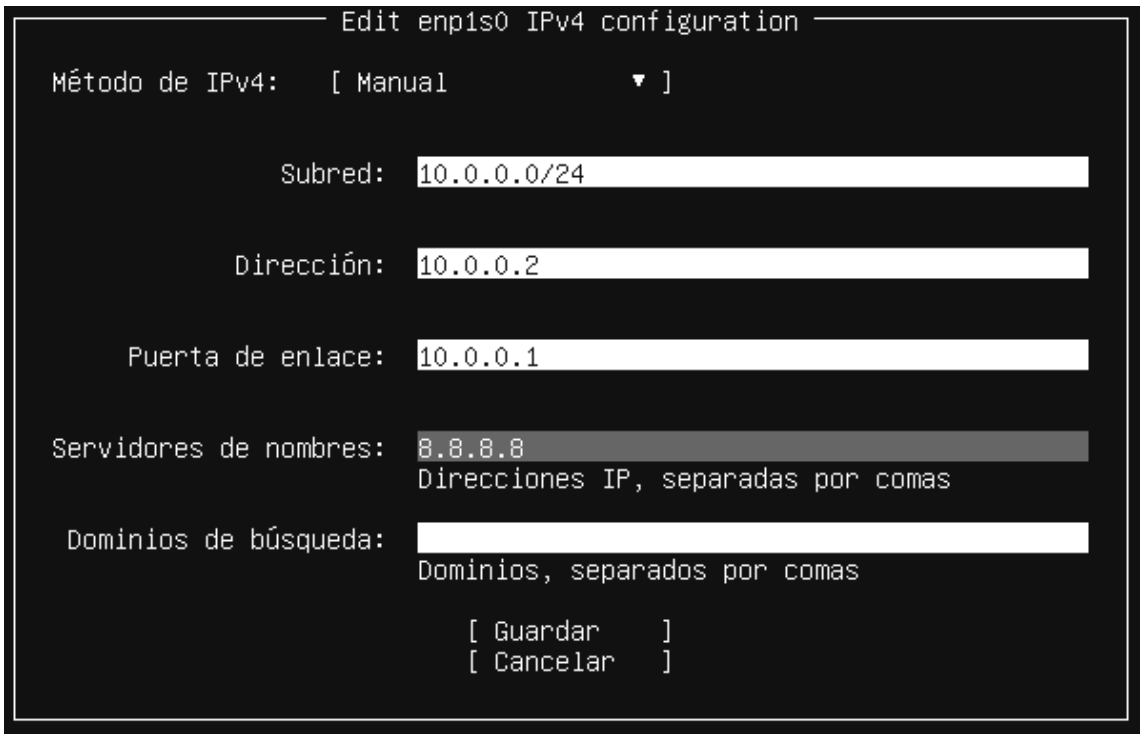

Ilustración 11. Asignación de IP

Anteriormente se le ha indicado a KVM el tamaño de cada una de las máquinas, ahora Ubuntu lo reconocerá lo aplicará automáticamente siguiendo la siguiente tabla:

Host name	**Tamaño (GB)**
Controller	150
Compute	150
Block Storage	100
OSM	200

Tabla 6. Almacenamiento de las máquinas.

Configuración SSH.

En este punto, lo que llevaremos a cabo es la preparación de los hosts para poder ser accedidos vía SSH por el host anfitrión (servidor). Entrando más en detalles, copiaremos la clave pública del servidor en la ruta /root/.ssh de las, máquinas host anfitrión. En caso de no poseer ningún par de claves en el servidor, deberemos crearlas por medio del comando keygen.

ssh-keygen -f "/home/mike/.ssh/known_hosts" -R "10.0.0.4"

A continuación, una vez hayamos podido acceder a la máquina vía ssh tendremos que copiar el contenido del siguiente archivo `/root/.ssh/authorized_keys` en el mismo directorio de los host.

La forma de acceder a cualquiera de las máquinas será mediante este comando:

```
# ssh NOMBRE_MÁQUINA@DIRECCIÓN_MÁQUINA
```

En el caso de la máquina Controller sería:

```
# ssh controller@10.0.0.2
```

Configuración de red.

Para poder configurar las redes de las máquinas, en primer lugar, tendremos que cerciorarnos de que los paquetes de red están instalados y actualizados [22]:

```
# apt-get install bridge-utils

# apt-get install ifupdown

# apt-get install net-tools
```

A continuación, abasteceremos las máquinas con su configuración de red apropiada rellenando los archivos /etc/network/interfaces de cada una de ellas.

Las configuraciones para este despliegue se encuentran en el Anexo B.

Configuración del despliegue

De vuelta en el host de despliegue, donde anteriormente hemos clonado los repositorios de openstack mediante la herramienta Git, copiaremos el contenido del directorio

/opt/openstack-ansible/etc/openstack_deploy en la ruta /etc/openstack_deploy Todo ello en el host de despliegue.

Entorno de configuración

El siguiente paso será modificar el archivo principal que dará forma a nuestra arquitectura. Aquí le indicaremos a ansible toda la información que necesita para desplegar el entorno: las redes disponibles, las máquinas disponibles, los servicios que queremos desplegar y por último, también le indicaremos específicamente las maquinas (dentro de las que le hemos declarado) donde queremos que se despliegue cada servicio en función del rol de la misma. Sin más dilación, copiamos el contenido del Anexo C un nuevo archivo que crearemos en la siguiente ruta: /etc/openstack_deploy/openstack_user_config.yml.

Credenciales

Una parte importante del proceso es garantizar la seguridad del sistema y para ello, este repositorio ya viene preparado, proporcionándonos la herramienta (en este caso un script) para poder generar un archivo de claves aleatorias del cual se nutrirá todo el sistema. Este script generará todas las credenciales tanto de acceso como de validación de todos los servicios, tanto de los desplegados como de los no desplegados y le indicaremos que las almacene en el archivo que nos indica la documentación oficial.

```
# cd /opt/openstack-ansible

#./scripts/pw-token-gen.py --file
/etc/openstack_deploy/user_secrets.yml
```

En nuestro caso, el archivo generado se encuentra en el Anexo H.

Ejecución de playbooks

En este apartado, ejecutaremos los playbooks de instalación. Cada uno de ellos realizará una serie de configuraciones y modificaciones en los hosts.

Setup-hosts.yml

Prepara los hosts para la infraestructura y los servicios Openstack, crea, configura con los requisitos necesarios y reinicia los contenedores que albergarán los servicios.

Setup-infrastructure.yml

Instala servicios de los cuales se compone la infraestructura: Memcached, el servidor de repositorio, Galera, RabbitMQ y rsyslog.

Setup-openstack.yml

Instala los servicios de Openstack, incluidos Identity (keystone), Image (look), Block Storage (cinder), Compute (nova), Networking (neutron), etc.

Ejecutamos:

```
# openstack-ansible setup-hosts.yml
```

La salida que esperamos tendrá la siguiente estructura:

```
PLAY                                                                    RECAP
*********************************************************************
****

...

deployment_host:  ok=18   changed=11   unreachable=0    failed=0
```

Es importante que antes de pasar al siguiente playbook, nos aseguremos de que no ha habido ningún task failed o unreachable.

```
# openstack-ansible setup-infrastructure.yml
```

La salida esperada será del mismo estilo que la anterior.

Finalmente, ejecutamos:

```
# openstack-ansible setup-openstack.yml
```

Una vez finalizado el proceso, podremos acceder a la API introduciendo la dirección IP del host controller.

En este punto obtendremos la siguiente interfaz:

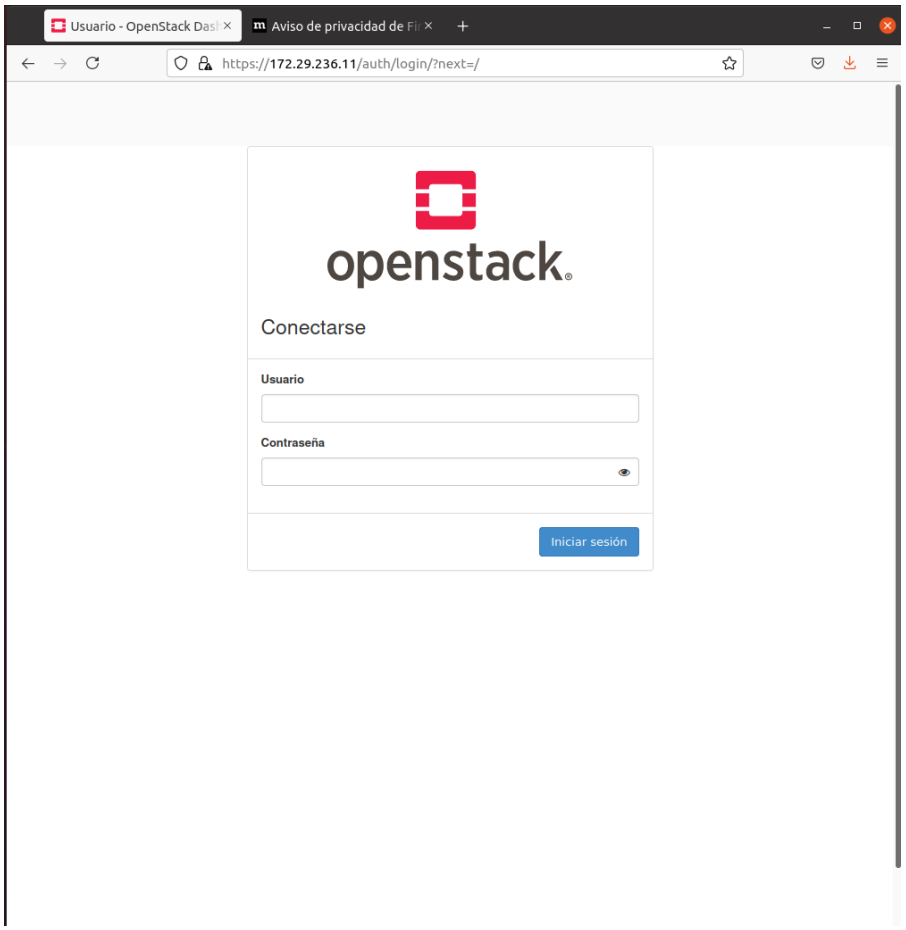

Ilustración 12. Página de Login de Openstack.

Para acceder nos solicita unas credenciales, el usuario será *admin* y la contraseña la obtendremos del archivo de credenciales que hemos configurado anteriormente y que se encuentra en el Anexo H:

```
keystone_auth_admin_password: 7d952e2c37d6d586edeebb7cd4487935e
```

Introduciremos el código sin el primer espacio.

Una vez autenticados, esta es la página que se nos mostrará en el navegador:

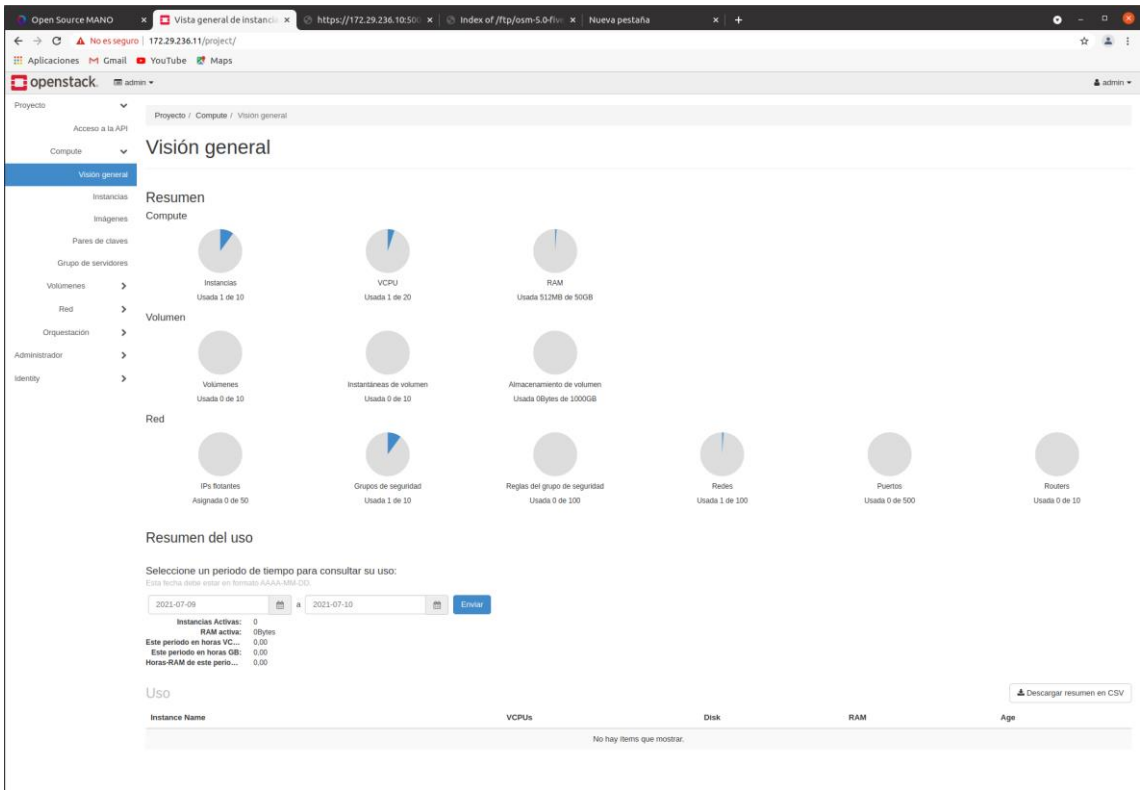

Ilustración 13. Interfaz de Openstack.

Llegados a este punto, tenemos pleno acceso a la interfaz y por lo tanto, a todos los servicios desplegados de Openstack. Ahora, estamos preparados para proceder con la integración de ambos servicios, OSM y Openstack.

En este apartado, Se produjo un error crítico que interrumpía el despliegue de la infraestructura durante la ejecución del playbook “setup-openstack.yml”. Tras una investigación exhaustiva, finalmente, y a pesar de no haber encontrado apenas información acerca de dicho error con la información que se tenía en ese momento, se consiguió encontrar una explicación y lo que es más importante la solución. La causa del error fue la no declaración de uno de los servicios en el archivo “openstack_user_config.yml”. En uno de los foros de desarrolladores pude constatar que los servicios nova y placement de Openstack habían dejado de trabajar conjuntamente por lo que este último servicio, placement, tenía que ser declarado en el archivo anteriormente mencionado. Esta información por alguna razón no venía recogida en la documentación oficial del despliegue y resulta vital para el buen funcionamiento del mismo.

En el Anexo D viene recogida la salida del error, así como la URL de una “disputa” abierta, en la cual aporté mi solución, en uno de los foros principales en el cual encontré un grupo de usuarios con el mismo problema y a los cuales nadie les había ofrecido una solución viable.

6. Integración de ambas plataformas

Proceso de integración

Previamente, antes de poder usar la terminal de la máquina controller, debemos llevar a cabo la exportación de variables globales en dicha máquina [23]. Esta tarea la ejecutaremos por medio de un script que nos ofrece la herramienta Openstack tanto en la app y web como por medio de la terminal de la máquina controller:

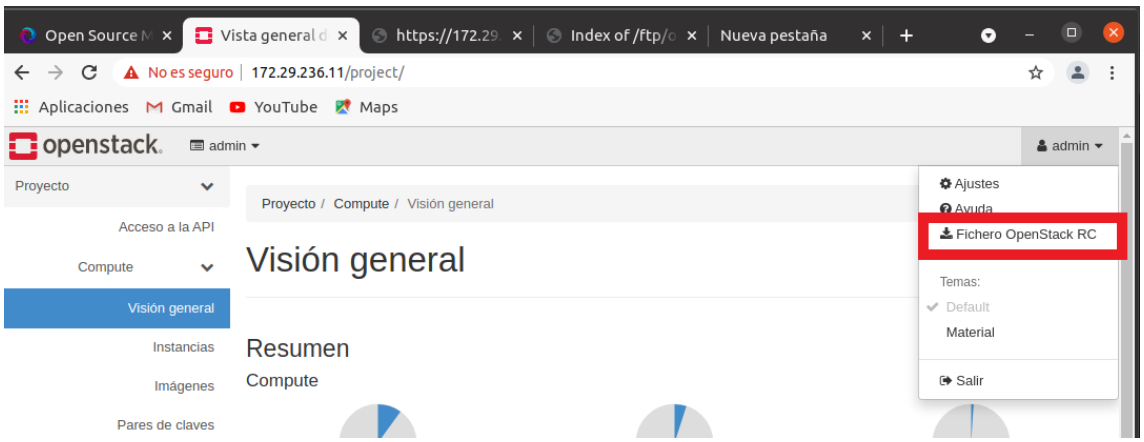

Ilustración 14. Localiación del fichero RC.

El contenido de dicho fichero se encuentra en el Anexo E.

Crearemos un archivo en nuestra máquina llamado admin.sh y copiaremos en él el contenido del archivo que hemos mencionado.

A continuación, ejecutaremos:

```
# source admin-openrc.sh
```

Seguimos instalando el paquete que nos permitirá interactuar con la plataforma:

```
# apt install python3-openstackclient
```

Ya tenemos nuestra terminal configurada para operar con Openstack y lo comprobaremos ejecutando el comando que se muestra a continuación y cerciorándonos de que nos devuelve la lista de servicios desplegados.

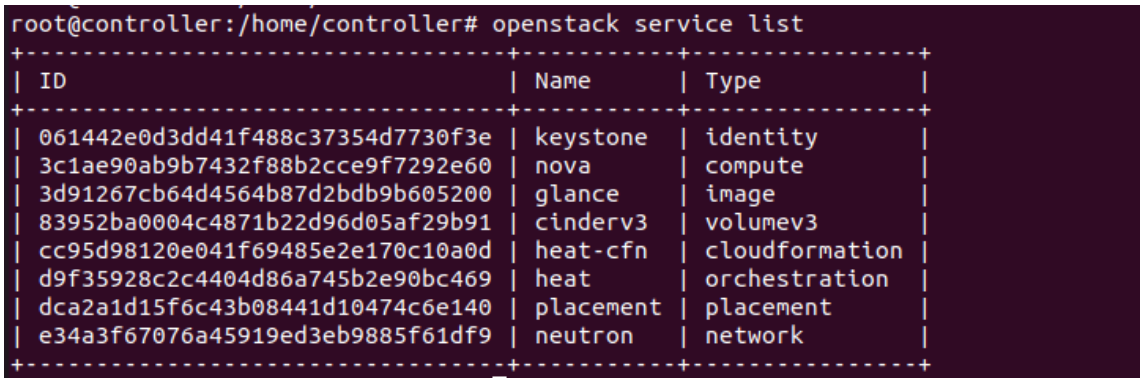

Ilustración 15. Comprobación de los servicios desplegados de Openstack.

Finamente, una vez hemos desplegado y comprobado el correcto funcionamiento de las infraestructuras por separado llega el momento de llevar a cabo la integración de ambas para que puedan trabajar de forma conjunta. tenemos dos principales formas para realizar dicha integración. Por un lado, podemos hacer uso de la interfaz gráfica que nos ofrece Open Source MANO accediendo mediante el navegador y con las credenciales que anteriormente hemos especificado. Y, por otro lado, podemos realizar la misma acción mediante la terminal de la máquina en la que se encuentra desplegado Open Source mano con el siguiente comando:

```
osm vim-create --name openstack-site --user admin --password
7d952e2c37d6d586edeebb7cd4487935e --auth_url
https://172.29.236.10:5000/v3/ --tenant admin --account_type
openstack --config '{insecure: True}'
```

En nuestro caso las credenciales que hemos usado para vincular ambas plataformas han sido las que openstack crea de serie durante el proceso de instalación. La contraseña, por otro lado, será de nuevo la asociada al usuario admin. Esta contraseña (keystone_auth_admin_password: 7d952e2c37d6d586edeebb7cd4487935e) se encuentra en el archivo de credenciales que hemos configurado en el apartado *Credenciales* del punto 7.

Para obtener la URL de autenticación tendremos que acudir a la interfaz web de openstack y pincharemos en la sección acceso a la API que se encuentra situada en la parte superior izquierda:

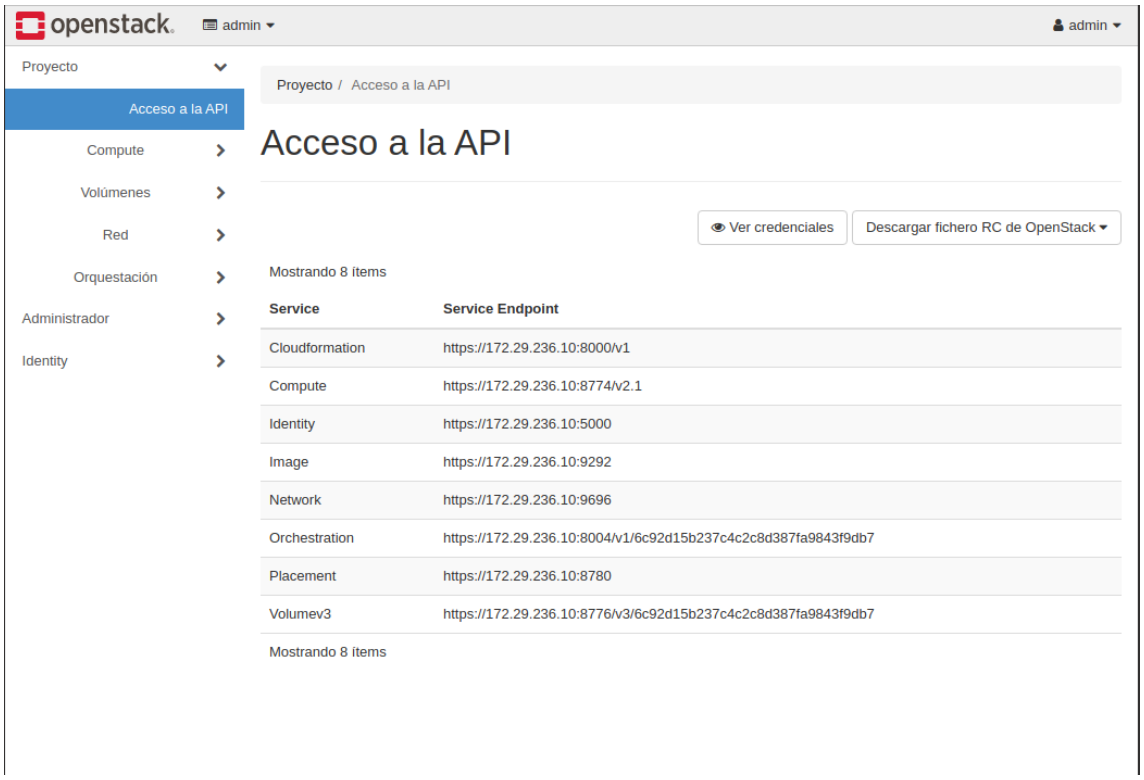

Ilustración 16. Localiación de la dirección URL del servicio Identity.

La dirección que usaremos para la integración será la asociada al servicio Identity. para comprobar la validez de esta dirección podemos introducirla en el navegador obteniendo una salida como la que se muestra a continuación:

Ilustración 17. Visualización de la URL de Identity.

Una vez realizada una de estas dos acciones, de nuevo, podemos comprobar la integración tanto por la interfaz gráfica como por la terminal de la misma máquina. A continuación, se muestran las salidas de ambas posibilidades:

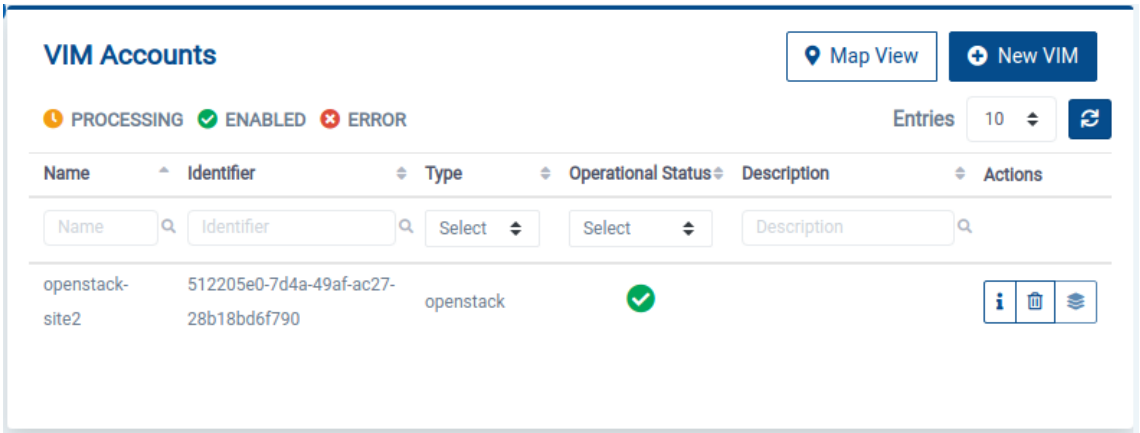

Ilustración 18. Comprobación del estado de la cuenta VIM en la API.

```
root@osm:/home/osm# osm vim-list
+-----------------+--------------------------------------+--------------------+
| vim name        | uuid                                 | operational state  |
+-----------------+--------------------------------------+--------------------+
| openstack-site2 | 512205e0-7d4a-49af-ac27-28b18bd6f790 | ENABLED            |
+-----------------+--------------------------------------+--------------------+
root@osm:/home/osm#
```

Ilustración 19. Comprobación del estado de la cuenta VIM en la terminal.

En este punto obtuvimos un error a la hora de ejecutar el comando de integración de ambas plataformas. Una vez más se llevó a cabo un trabajo de diagnóstico del error sin llegar a encontrar una solución clara al mismo (las salidas de los logs se encuentran en el Anexo F), finalmente pudimos identificar que se trataba de un error del protocolo SSL y que por alguna razón el sistema no se estaba comportando cómo se espera. La solución temporal adoptada fue utilizar las opciones apropiadas de este comando para eludir la comprobación de los certificados SSL y así poder continuar con nuestro despliegue hasta que haya una solución oficial al problema. Esta información fue puesta en conocimiento de los desarrolladores de Open Source MANO y publicada en los foros dónde otros usuarios preguntaban por el mismo problema. En el Anexo F se encuentra la URL de mi solución planteada.

Configuración y abastecimiento para el trabajo conjunto

Openstack

Fundamentalmente, son cuatro las necesidades que tendremos que satisfacer para poder hacer un uso básico de los servicios de openstack. En todos los casos que

repasaremos en este apartado se intentará mostrar la versión vía API tanto como la vía haciendo uso de la terminal.

Creación de una red

Tenemos dos formas para efectuar el proceso de creación de una red. Por un lado, podemos operar haciendo uso de la API, en la cual seleccionaremos las pestañas Proyecto -> Red -> Redes y seleccionaremos la casilla *Crear una red.*

A continuación, se muestra una posible configuración para una red con su correspondiente subnet dentro de openstack:

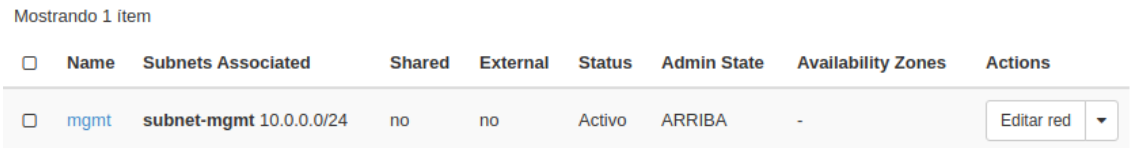

Ilustración 20. Creación de una red.

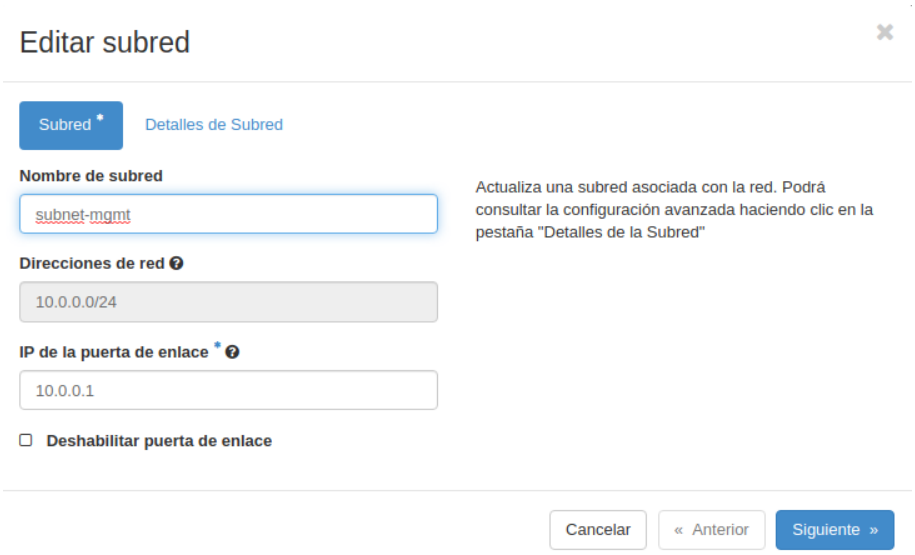

Ilustración 21. Creación de una subred.

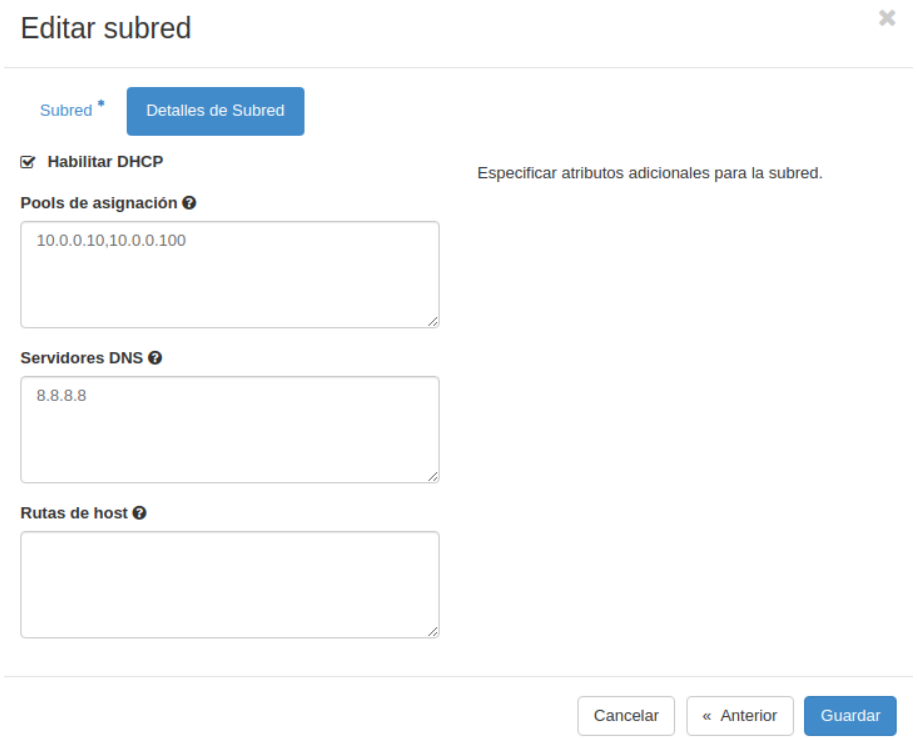

Ilustración 22. Especificación de los detalles de la subred.

Mediante La terminal de la máquina controller procederíamos de la siguiente manera:

```
root@controller:/etc# neutron net-create mgmt --provider:network_type=vlan --
neutron CLI is deprecated and will be removed in the future. Use openstack CLI i
nstead.
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2021-06-28T16:17:01Z                 |
| description               |                                      |
| id                        | c8b3994b-04b7-436c-bbdf-d9f0e06bcde4 |
| ipv4_address_scope        |                                      |
| ipv6_address_scope        |                                      |
| is_default                | False                                |
| mtu                       | 1500                                 |
| name                      | mgmt                                 |
| port_security_enabled     | True                                 |
| project_id                | 2bd724a2987046978ecdd2c4394eb97d     |
| provider:network_type     | vlan                                 |
| provider:physical_network | vlan                                 |
| provider:segmentation_id  | 148                                  |
| revision_number           | 1                                    |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | 2bd724a2987046978ecdd2c4394eb97d     |
| updated_at                | 2021-06-28T16:17:01Z                 |
+---------------------------+--------------------------------------+
```

Ilustración 23. Creacion de una red en la terminal.

```
root@controller:/etc# neutron subnet-create --name subnet-mgmt mgmt 10.208.0.0/2
4 --allocation-pool start=10.208.0.2,end=10.208.0.254
neutron CLI is deprecated and will be removed in the future. Use openstack CLI i
nstead.
Created a new subnet:
+-------------------+--------------------------------------------------+
| Field             | Value                                            |
+-------------------+--------------------------------------------------+
| allocation_pools  | {"start": "10.208.0.2", "end": "10.208.0.254"}   |
| cidr              | 10.208.0.0/24                                    |
| created_at        | 2021-06-28T16:20:57Z                             |
| description       |                                                  |
| dns_nameservers   |                                                  |
| enable_dhcp       | True                                             |
| gateway_ip        | 10.208.0.1                                       |
| host_routes       |                                                  |
| id                | a9fdb65c-4126-41ed-adad-90eb7eb83276             |
| ip_version        | 4                                                |
| ipv6_address_mode |                                                  |
| ipv6_ra_mode      |                                                  |
| name              | subnet-mgmt                                      |
| network_id        | c8b3994b-04b7-436c-bbdf-d9f0e06bcde4             |
| project_id        | 2bd724a2987046978ecdd2c4394eb97d                 |
| revision_number   | 0                                                |
| service_types     |                                                  |
| subnetpool_id     |                                                  |
| tags              |                                                  |
| tenant_id         | 2bd724a2987046978ecdd2c4394eb97d                 |
| updated_at        | 2021-06-28T16:20:57Z                             |
+-------------------+--------------------------------------------------+
```

Ilustración 24. Creación de una subred en la terminal.

Creación de un par de claves

Es importante la creación de un par de claves para más tarde poder tener acceso vía SSH a las máquinas. Para crear dicho par de claves desde la API, nos desplazaremos a Proyecto -> Compute -> Pares de claves y pincharemos en *Crear Par De Claves.* En cambio, si no tenemos acceso a la API, lo podremos hacer mediante el siguiente comando:

```
root@controller:/home/controller# openstack keypair create test2
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA1GpEeBSokYp9Ko+fAQ/hXOys3DSZjBAQjCf8y3OE1uAJHJZj
I2nZJh7YW3BuwrYe6X6hyBY7Th08KWAnKaU90izQdg7F8NAAKVFFiYsVvhv+yxhS
JTB+tG1r6kIcTLlldqgudigutlgNNUzCOquIDyePkcFPjHDsKIL8r6ZfmxNZHwyQ
7mrL7QkTvTbF6GRYsaRdNyHg3fJ+1p0MtvcnWEHs97wCL/dxg24Fp6MIPTVLw06Z
Hdl+b1itXCHRFitpZ8bxrfmBiczeHfl/dJu3mR0cwmfwlJAXrMcZjLtDgv4IPEu8
3mDRfnRozd/hRrx7bvR08qGQJhrvON82JfoO3QIDAQABAoIBACfIJz6eD9wCJeL8
tvjJ8pkmslEjgMZck9zL+GpaHjzoBIzYFYUiY8rh5/9oFTZeSXfNgZspnr3Id+X/
3hXMnsFoJzqVFbk+G2mhkPZrmAzczWFYErpoVLKR864R557u8fPBF8KFUApSf1bG
I3I9gePkstzHaZCR7YjAz3ZOIIW4SIlwDXsX9MTBFgOuE8YfUJ9k588mg8XVmjKg
XOCH8U6P+07Q0vp/a6J0OWH/JWIOWU73ej6N6QOg2c7iInVl2S+N0URoFTUTS5Sn
DPHyoHd/vOZq6cdg7uD0781NmbKMTI1ICDKGuwwVmZ9vWCM+nlPP3LYCZ/o4LAT/
eXY0sgECgYEA8RJR2s+3Eu7AU7os3g2Dxy2LlpOJEZGTuvukHxdzhtsjP7SRxQo+
B+w1ljbp3MydHi06lyybN9PwYTfHngYBKR2ShBoXcAist9bsusKzP4TszNoJSAX5
8STACG/qkhAtRr+cWRG9m8CylZxUU+K0sQF6lP03lao83fyqbBX1tcECgYEA4ZGp
AWPgPAtiriRoe6+YzfHwqdrnNpNMHj7IYhy5G55W06oc5QT4ZewnS0+stcfdupgw
bnC39xFWFIA1S881qyyRGY+EQt6uFtgpDxedwBMu7WAilfzyisXaFUdq+RxCMgpq
8p37UbbnCkbB6jFKSsXyqxFW1PhBjPoO3IxyeB0CgYBZbAg8+EvalAF3jVnTTIQT
QAvFYRi5EZ0RAQMSHWrXHTZoDRz9UB1OZwpBnEEHjEbDyNcw+HTjMpS6GyzKq0OG
8tdEdUE4Z59Z2jmZ1VsUNsR4AHDs7HqYPp+VAZ6ra6C/25etWm1Q9xvHb2PAchnA
huHnnj4eIDXPykeVLFgmgQKBgDgnZdZlwb3Q68kEYUphRX4He+0dBLBObQWehY3/
LonUmjLpR3YwDZJ5/rI97HX4gpdgKbZkszd7hl1mKV5tACRT3O9rDfcDrn0bZOVP
8MKPRABuDU4fgVXQuWDQva20Pjps44DwmU8XZF7njIC7so34x1eU4nP0NYVd/vMi
t6B1AoGAU/rzNoJhwGsi/FHgWHX+uysl1x4FHRXZUvSdWNEtbQxRe7KHulbqfPCB
Ttj3sFay9khDU8hnHfJ116Zkbij+xycjHJyvkuobvMc59GKx/ibe976WvbiACmFp
ofoQ14BGAcvbvKjxe7ZucrgCMFeT7XybaK1PwRRDzEeQ2Y3epfY=
-----END RSA PRIVATE KEY-----

root@controller:/home/controller#
```

Ilustración 25. Creación de una par de claves.

Establecimiento de un flavour

Antes de nada, un flavour no es más que definición de las especificaciones básicas que va a tener un prototipo especifico de instancia. Para añadir un flavour vía API procederemos de la siguiente manera, nos desplazaremos a Administrador -> Compute -> Sabores -> Crear sabor.

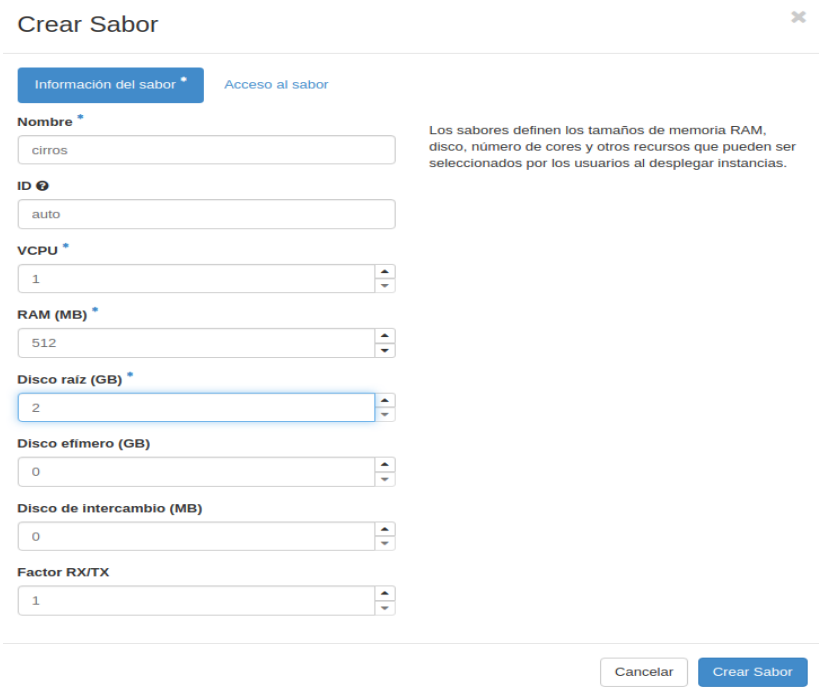

Ilustración 26. Creación de Flavour.

Mediante la terminal bastaria con ejecutar el siguiente comando:

```
root@controller:/home/controller# openstack flavor create ubuntu --id ubuntu --ram 1 --disk 10 --vcpus 1
+----------------------------+--------+
| Field                      | Value  |
+----------------------------+--------+
| OS-FLV-DISABLED:disabled   | False  |
| OS-FLV-EXT-DATA:ephemeral  | 0      |
| disk                       | 10     |
| id                         | ubuntu |
| name                       | ubuntu |
| os-flavor-access:is_public | True   |
| properties                 |        |
| ram                        | 1      |
| rxtx_factor                | 1.0    |
| swap                       |        |
| vcpus                      | 1      |
+----------------------------+--------+
root@controller:/home/controller#
```

Ilustración 27. Creación de un Flavour mediante la terminal.

Creación de una imagen

Para subir una imagen, lo primero que necesitaremos es la propia imagen del sistema operativo que queramos emplear. En nuestro caso, se ha empleado una imagen de Cirros (http://download.cirros-cloud.net/0.5.1/cirros-0.5.1-x86_64-disk.img).

En el caso de la API, se procederá de la siguiente manera, nos desplazamos a Proyecto -> Compute -> Imágenes -> Crear imagen.

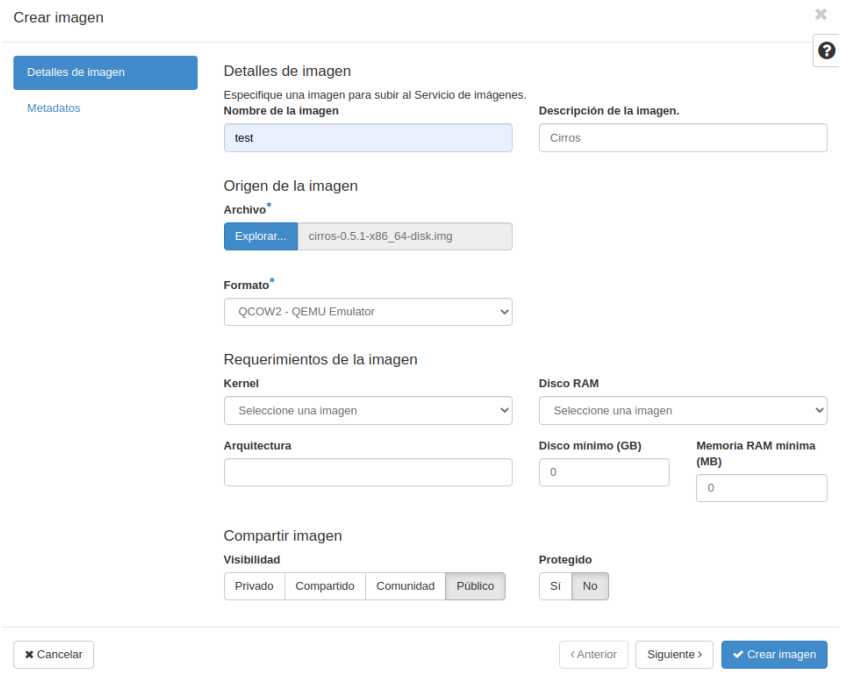

Ilustración 28. Creación de una imagen.

En el caso de querer proceder mediante la terminal ejecutaremos el siguiente comando:

```
root@controller:/home/controller# openstack image create --file
./cirros-0.3.4-x86_64-disk.img --container-format=bare --disk-
format=qcow2 --public cirros034
```

Open Source Mano

Para poder realizar órdenes de creación de servicios de red desde Open Source MANO son imprescindibles los descriptores VNFD y NSD. En nuestro caso hemos podido utilizar unos de los descriptores que ofrece Open Source MANO en su documentación oficial (https://osm-download.etsi.org/ftp/Packages/examples/).

Para incluir los descriptores en la plataforma bastaría con arrástralos a la zona que nos indica la API web. Para el caso de los paquetes NS:

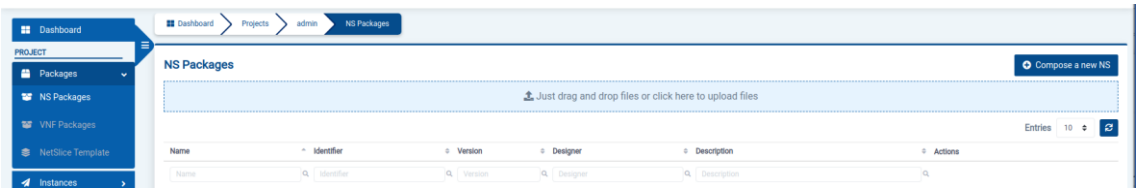

Ilustración 29. Creación de paquetes NS.

Y para el caso d ellos paquetes VNF:

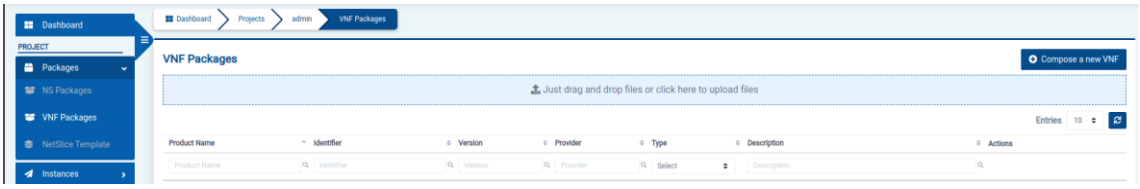

Ilustración 30. Creación de paquetes VNF.

7. Comprobación de resultados

Una vez desplegadas las dos herramientas, configurado cada una por separado y finalmente habiendo integrado ambas en una única plataforma que podrá ser controlada y gestionada desde Open Source MANO, procedemos a instanciar una máquina desde esta última plataforma.

Para ello, nos iríamos a Instances -> NS Instances -> New NS:

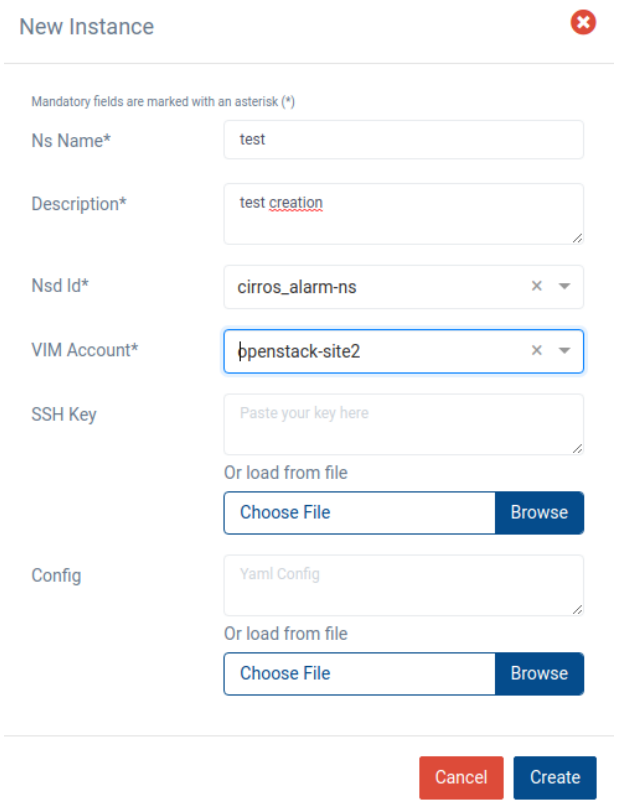

Ilustración 31. Lanzamiento de una instancia en OSM.

Pasados unos segundos, aparece un error en la interfaz de OSM. Si nos dirigimos a Openstack nos encontramos un error en el proceso de instanciación tal y como se muestra a continuación:

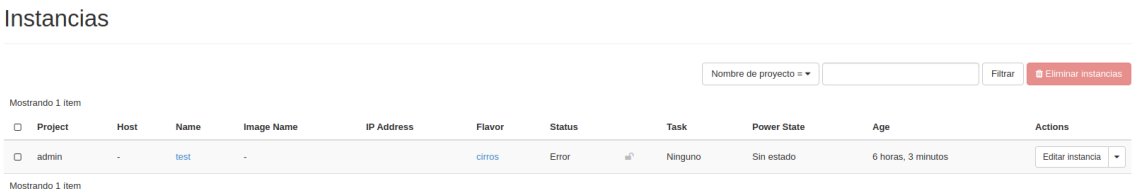

Ilustración 32. Estado de la instancia lanzada en Openstack.

Una vez detectado, los siguientes pasos fueron recopilar la máxima cantidad de información para acotar el escenario de actuación de cara a su resolución. Por un lado, la propia API de Openstack, pinchando en la instancia, nos ofrecía la siguiente traza:

```
Mensaje

Build of instance 1de3b0d0-a26a-45b3-afa5-380ca813d091 aborted: Volume 1c3a995a-e259-4b92-94c2-3c1f62618e75 did not finish being created even after we waited 3 seconds or 2 attempts. And its status is error.

Código

500

Detalles

Traceback (most recent call last): File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/compute/manager.py", line 1976, in _prep_block_device driver_block_device.attach_block_devices( File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/virt/block_device.py", line 874, in attach_block_devices _log_and_attach(device) File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/virt/block_device.py", line 871, in _log_and_attach bdm.attach(*attach_args, **attach_kwargs) File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/virt/block_device.py", line 769, in attach self.volume_id, self.attachment_id = self._create_volume( File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/virt/block_device.py", line 384, in _create_volume self._call_wait_func(context, wait_func, volume_api, vol['id']) File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/virt/block_device.py", line 728, in _call_wait_func LOG.warning( File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/oslo_utils/excutils.py", line 227, in __exit__ self.force_reraise() File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/oslo_utils/excutils.py", line 200, in force_reraise
```

raise self.value File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/virt/block_device.py", line 721, in _call_wait_func wait_func(context, volume_id) File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/compute/manager.py", line 1591, in _await_block_device_map_created raise exception.VolumeNotCreated(volume_id=vol_id, nova.exception.VolumeNotCreated: Volume 1c3a995a-e259-4b92-94c2-3c1f62618e75 did not finish being created even after we waited 3 seconds or 2 attempts. And its status is error. During handling of the above exception, another exception occurred: Traceback (most recent call last): File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/compute/manager.py", line 2605, in _build_resources block_device_info = self._prep_block_device(context, instance, File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/compute/manager.py", line 1996, in _prep_block_device raise exception.InvalidBDM(str(ex)) nova.exception.InvalidBDM: Volume 1c3a995a-e259-4b92-94c2-3c1f62618e75 did not finish being created even after we waited 3 seconds or 2 attempts. And its status is error. During handling of the above exception, another exception occurred: Traceback (most recent call last): File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/compute/manager.py", line 2232, in _do_build_and_run_instance self._build_and_run_instance(context, instance, image, File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/nova/compute/manager.py", line 2441, in _build_and_run_instance compute_utils.notify_about_instance_create(File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/oslo_utils/excutils.py", line 227, in __exit__ self.force_reraise() File "/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-packages/oslo_utils/excutils.py", line 200, in force_reraise raise self.value File

```
"/openstack/venvs/nova-23.0.0.0rc1/lib/python3.8/site-
packages/nova/compute/manager.py", line 2392, in
_build_and_run_instance with self._build_resources(context,
instance, File "/usr/lib/python3.8/contextlib.py", line 113, in
__enter__ return next(self.gen) File "/openstack/venvs/nova-
23.0.0.0rc1/lib/python3.8/site-
packages/nova/compute/manager.py", line 2615, in _build_resources
raise exception.BuildAbortException(instance_uuid=instance.uuid,
nova.exception.BuildAbortException: Build of instance 1de3b0d0-
a26a-45b3-afa5-380ca813d091 aborted: Volume 1c3a995a-e259-4b92-
94c2-3c1f62618e75 did not finish being created even after we
waited 3 seconds or 2 attempts. And its status is error.
```

Analizando la traza, se puede observar que es un error en la conexión (Error 500, “El código de respuesta **`500 Error Interno del Servidor`** indica que el servidor encontró una condición inesperada que le impide completar la petición.” [24]). Tras ello, procedemos a comprobar los logs del sistema de la máquina Block Storage, ya que es l máquina que contiene el servicio que está fallando (Cinder) con el siguiente comando:

```
# journalctl -xe
```

La opción -xe nos permite filtrar las últimas entradas al log. [25]

La salida que nos devuelve dicho comando es la siguiente:

```
Jul 10 04:29:11 block cinder-volume[2394]: 2021-07-10
04:29:11.434 2394 ERROR cinder.service [-] Manager for service
cinder-volume block@lvm is reporting problems, not sending
heartbeat. Service will appear "down”.
```

La primera comprobación que se llevó a cabo fue el cerciorarnos de que la máquina Block Storage tenía conexión con las máquinas Controller y Compute. No tuvo efecto por lo que continuamos comprobando el estado de los volúmenes de almacenamiento:

```
root@controller:/home/controller# openstack volume service list
+------------------+-------------------------------------+------+---------+-------+----------------------------+
| Binary           | Host                                | Zone | Status  | State | Updated At                 |
+------------------+-------------------------------------+------+---------+-------+----------------------------+
| cinder-volume    | block@lvm                           | nova | enabled | down  | 2021-07-09T14:02:45.000000 |
| cinder-scheduler | infra1-cinder-api-container-eac33a27 | nova | enabled | up    | 2021-07-10T04:41:21.000000 |
+------------------+-------------------------------------+------+---------+-------+----------------------------+
root@controller:/home/controller#
```

Ilustración 33. Comprobación del estado de los servicios Cinder.

En la ilustración 31 comprobamos que, tal y como nos indicaba la salida del comando “journalctl -xe”, uno de los volúmenes de almacenamiento se encuentra en estado “down”.

La solución más inmediata encontrada en los foros de desarrolladores (ya que la documentación oficial no recoge este error) fue la de reiniciar el servicio cinder-volume en la máquina Block storage. Tampoco tuvo efecto, el resultado fue el mismo.

El siguiente paso fue reiniciar de nuevo el servicio cinder-volume para visualizar los diferentes estados o fases por los que pasa en su arranque, para de esta forma capturar algún mensaje o alguna traza que nos dé más información.

Se ejecutan los siguientes comandos para ello, todos ellos en la máquina Block Storage. Para reiniciar el servicio:

```
# systemctl restart cinder-volume
```

Y para visualizar su estado en la terminal:

```
# systemctl status cinder-volume
```

Una vez operativo el servicio, se obtiene la siguiente salida:

```
root@block:/etc/systemd/system# systemctl status cinder-volume
● cinder-volume.service - cinder-volume service
     Loaded: loaded (/etc/systemd/system/cinder-volume.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2021-07-10 09:27:15 UTC; 2min 31s ago
   Main PID: 12985 (cinder-volume)
      Tasks: 2 (limit: 3407)
     Memory: 104.9M
        CPU: 2min 24.102s
     CGroup: /cinder.slice/cinder-volume.service
             ├─12985 /openstack/venvs/cinder-23.0.0.0rc1/bin/python /openstack/venvs/cinder-23.0.0.0rc1/bin/cinder-volume
```

```
             └─13013                    /openstack/venvs/cinder-
23.0.0.0rc1/bin/python                  /openstack/venvs/cinder-
23.0.0.0rc1/bin/cinder-volume


Jul 10 09:28:55 block sudo[13024]:   cinder : TTY=unknown ; PWD=/
;       USER=root       ;       COMMAND=/openstack/venvs/cinder-
23.0.0.0rc1/bin/cinder-rootwrap   /etc/cinder/rootwrap.conf   env
LC_ALL=C vgs --version

Jul  10  09:28:55  block  sudo[13024]:  pam_unix(sudo:session):
session opened for user root by (uid=0)

Jul  10  09:29:12  block  sudo[13024]:  pam_unix(sudo:session):
session closed for user root

Jul 10 09:29:16 block sudo[13032]:   cinder : TTY=unknown ; PWD=/
;       USER=root       ;       COMMAND=/openstack/venvs/cinder-
23.0.0.0rc1/bin/cinder-rootwrap   /etc/cinder/rootwrap.conf   env
LC_ALL=C vgs --version

Jul  10  09:29:16  block  sudo[13032]:  pam_unix(sudo:session):
session opened for user root by (uid=0)

Jul  10  09:29:29  block  sudo[13032]:  pam_unix(sudo:session):
session closed for user root

Jul 10 09:29:32 block sudo[13039]:   cinder : TTY=unknown ; PWD=/
;       USER=root       ;       COMMAND=/openstack/venvs/cinder-
23.0.0.0rc1/bin/cinder-rootwrap   /etc/cinder/rootwrap.conf   env
LC_ALL=C vgs --noheadings -o name ci>

Jul  10  09:29:32  block  sudo[13039]:  pam_unix(sudo:session):
session opened for user root by (uid=0)

Jul  10  09:29:44  block  sudo[13039]:  pam_unix(sudo:session):
session closed for user root

Jul   10   09:29:46   block   cinder-volume[13013]:   2021-07-10
09:29:44.615  13013  ERROR  cinder.volume.manager  [req-8e8405fa-
b37d-44ab-9fd4-389215c9e9b9  -  -  -  -  -]  Failed  to  initialize
driver.: oslo_concurrency.processu>
```

Command: sudo cinder-rootwrap /etc/cinder/rootwrap.conf env LC_ALL=C vgs --noheadings -o name cinder-volumes

Exit code: 5

Stdout: ''

Stderr: ' Volume group "cinder-volumes" not found\n Cannot process volume group cinder-volumes\n'

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager Traceback (most recent call last):

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager File "/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/cinder/volume/manager.py", line 481,>

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager self.driver.check_for_setup_error()

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager File "/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/cinder/volume/drivers/lvm.py", line >

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager self.vg = lvm.LVM(

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager File "/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/cinder/brick/local_dev/lvm.py", line>

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager if self._vg_exists() is False:

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager File "/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/cinder/brick/local_dev/lvm.py", line>

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager (out, _err) = self._execute(*cmd,

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager File "/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/os_brick/executor.py", line 52, in _>

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager result = self.__execute(*args, **kwargs)

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager File "/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/cinder/utils.py", line 114, in execu>

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager return processutils.execute(*cmd, **kwargs)

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager File "/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/oslo_concurrency/processutils.py", l>

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager raise ProcessExecutionError(exit_code=_returncode,

2021-07-10 09:29:44.615 13013 ERROR cinder.volume.manager oslo_concurrency.processutils.ProcessExecutionError: Unexpected error while running command.

```
                                                    2021-07-10
09:29:44.615 13013 ERROR cinder.volume.manager Command: sudo
cinder-rootwrap /etc/cinder/rootwrap.conf env LC_ALL=C vgs --
noheadings -o name cinder-volumes

                                                    2021-07-10
09:29:44.615 13013 ERROR cinder.volume.manager Exit code: 5

                                                    2021-07-10
09:29:44.615 13013 ERROR cinder.volume.manager Stdout: ''

                                                    2021-07-10
09:29:44.615 13013 ERROR cinder.volume.manager Stderr: '  Volume
group "cinder-volumes" not found\n  Cannot process volume group
cinder-volumes\n'

                                                    2021-07-10
09:29:44.615 13013 ERROR cinder.volume.manager
```

En este punto, se nos indica que el error se está produciendo en los siguientes archivos:

/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/cinder/volume/manager.py (Línea 481)

/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/cinder/volume/drivers/lvm.py (Líneas 312, 115 y 146)

/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/os_brick/executor.py (Línea 52)

/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/cinder/utils.py (Línea 114)

/openstack/venvs/cinder-23.0.0.0rc1/lib/python3.8/site-packages/oslo_concurrency/processutils.py (Línea 438)

En el foro de desarrolladores [26] encontramos varios usuarios que notifican el mismo problema. No obtienen respuesta.

Tras una larga búsqueda, investigación y numerosos intentos para resolver el problema, finalmente no se consigue arreglar. El diagnóstico final ha reflejado que la comunicación entre Open Source MANO y Openstack es correcta ya que la llamada ha generado la

acción de creación de la instancia en Openstack. Este mismo procedimiento de creación se replicó desde Openstack directamente sin usar Open Source MANO de intermediario y el resultado fue el mismo. Por lo que se puede constatar que el problema se encuentra al margen de la integración entre ambas plataformas. El problema se produce cuando el sistema lleva a cabo la tarea de asignación espacio de almacenamiento a la hora de crear una instancia, es en ese punto donde falla y saltan las excepciones de los archivos Python que se señalan anteriormente.

8. Conclusiones

Este trabajo nos ha permitido conocer la forma en la que operan dos de las plataformas de administración de red que más futuro tienen, así como procesos de automatización con un gran potencial que permiten un abastecimiento y configuración de máquinas ofreciendo un alto nivel de abstracción al usuario.

Las tecnologías de virtualización son claramente una tendencia a futuro y este TFG se argumenta cada uno los beneficios encontrados en ellas.

Uno de los objetivos era automatizar el despliegue de ambas plataformas y se ha conseguido.

En última instancia nos hemos visto limitados por un error que finalmente no se ha podido resolver, pero creo que el trabajo realizado ha sido bastante productivo, así como útil para la comunidad que sustenta ambas plataformas, dando solución de algunos errores o bugs y por supuesto, contribuyendo con la aportación de las soluciones tomadas en cada caso.

Otro objetivo era hacer de este TFG un riguroso manual de despliegue que prestase atención a cada mínimo detalle durante el proceso. Desde mi punto de vista también se ha conseguido, ya que ofrece una serie de procedimientos y soluciones a errores encontrados, los cuales no se ha podido encontrar precedentes de subsanación.

Uno de los trabajos más duros de este proyecto ha sido sin duda la formación y el aprendizaje sobre estas plataformas, en especial Openstack. La curva de aprendizaje en el caso de este último es realmente pronunciada, generando habitualmente situaciones de confusión o incertidumbre.

Por otro lado, he encontrado cierto nivel de abstracción en la documentación oficial de Openstack, llegando a ser muy genérica en algunos aspectos. Este hecho ha dificultado en cierto modo este despliegue.

Finalmente, otro objetivo principal era el aprendizaje sobre las tecnologías de virtualización de red y sin duda se ha conseguido, llegando a tener una amplia comprensión del entorno así de cómo se relacionan entre sí los servicios que se despliegan en su interior.

9. Bibliografía

[1] L.Doyle, «¿Qué es NFV y cuáles son sus costes, rendimiento y beneficios de escala?,» 10 Feb 2018. [En línea]. Disponible: https://www.networkworld.es/movilidad/que-es-nfv-y-cuales-son-sus-costes-rendimiento-y-beneficios-de-escala. [Último acceso: 9 06 2021].

[2] SDxCentral Studios, «Sdxcentral,» 9 10 2014. [En línea]. Disponible: https://www.sdxcentral.com/networking/nfv/mano-lso/definitions/nfv-mano/. [Último acceso: 19 06 2021].

[3] A. Gallego., «Data-Driven resource orchestration in sliced 5G Networks,» Escuela Politécnica de Alcalá. Alcalá de Henares. España. , 2019.

[4] J. English., «NFV MANO,» Oct 2016. [En línea]. Disponible: https://searchnetworking.techtarget.com/definition/NFV-MANO-network-functions-virtualization-management-and-orchestration. [Último acceso: 13 07 2021].

[5] OSM, «OSM ETSI,» 2021. [En línea]. Disponible: https://osm.etsi.org/docs/user-guide/03-installing-osm.html. [Último acceso: 15 07 2021].

[6] OpenStack, «Overview,» 2019. [En línea]. Disponible: https://docs.openstack.org/newton/install-guide-ubuntu/overview.html. [Último acceso: 25 07 2021].

[7] S. Nangare, «OpenStack and Open Source MANO: Technologies for NFV Deployment,» 8 8 2018. [En línea]. Disponible: https://devops.com/openstack-and-open-source-mano-technologies-for-nfv-deployment/. [Último acceso: 17 06 2021].

[8] Red Hat, «Ansible,» 2021. [En línea]. Disponible: https://www.ansible.com/. [Último acceso: 3 6 2021].

[9] «Red Hat,» 2020. [En línea]. Disponible: https://www.redhat.com/es/topics/automation/learning-ansible-tutorial. [Último acceso: 12 07 2021].

[10] L. D. Barrio, «luisiblogdeinformatica,» 29 6 2020. [En línea]. Disponible: https://luisiblogdeinformatica.com/tutorial-ansible-desde-0-herramienta-de-gestion-de-servidores/. [Último acceso: 21 06 2021].

[11] A. B. Vicente, «Estudio de la plataforma de,» 2018.

[12] «Telco Systems,» 19 04 2021. [En línea]. Disponible: https://www.telco.com/knowledge_center/sdxcentral-arm-has-the-potential-to-disrupt-nfvi-market/. [Último acceso: 19 06 2021].

[13] D. Ruiz., «DISEÑO E IMPLEMENTACIÓN DE PLATAFORMA PARA EXPERIMENTACIÓN EN NFV,» Escuela de Ingeniería de Bilbao. Bilbao. España., 2019.

[14] ETSI., «Open Source MANO,» 2021. [En línea]. Disponible: https://www.etsi.org/technologies/open-source-mano. [Último acceso: 2 7 2021].

[15] J. G. Sotoca, «Sotoca,» 12 07 2014. [En línea]. Disponible: https://sotoca.es/category/openstack/. [Último acceso: 19 07 2021].

[16] Openstack, «Openstack,» 29 04 2021. [En línea]. Disponible: https://docs.openstack.org/openstack-ansible/queens/user/test/example.html. [Último acceso: 19 7 2021].

[17] Open Source MANO, «Open Source MANO,» 2021. [En línea]. Disponible: https://osm.etsi.org/docs/user-guide/03-installing-osm.html#checking-your-installation. [Último acceso: 26 7 2021].

[18] «Ubuntu Manuals,» 2019. [En línea]. Disponible: http://manpages.ubuntu.com/manpages/xenial/es/man8/apt-key.8.html. [Último acceso: 30 06 2021].

[19] X. AZNAR, «onthedock,» 14 04 2017. [En línea]. Disponible: https://onthedock.github.io/post/170414-error_the-connection-to-the-server-was-refused/. [Último acceso: 13 06 2021].

[20] «Zona System,» 6 2 2019. [En línea]. Disponible: https://www.zonasystem.com/2019/02/habilitar-virtualizacion-anidada-vt-x-amd-v-virtualbox.html. [Último acceso: 26 7 2021].

[21] Openstack, «Openstack,» 14 07 2021. [En línea]. Disponible: https://docs.openstack.org/project-deploy-guide/openstack-ansible/wallaby/targethosts.html. [Último acceso: 15 07 2021].

[22] L. Rendek, 26 11 2020. [En línea]. Disponible: https://linuxconfig.org/how-to-switch-back-networking-to-etc-network-interfaces-on-ubuntu-20-04-focal-fossa-linux. [Último acceso: 9 06 2021].

[23] A. Battisti, «Youtube,» 10 10 2020. [En línea]. Disponible: https://www.youtube.com/watch?v=Nwp7NGAKXA4&t=768s&ab_channel=AnselmoBattisti. [Último acceso: 12 07 2021].

[24] MDN Web Docs, «MDN Web Docs,» 16 07 2021. [En línea]. Disponible: https://developer.mozilla.org/es/docs/Web/HTTP/Status/500. [Último acceso: 21 07 2021].

[25] «Conpilar,» 6 03 2021. [En línea]. Disponible: https://conpilar.es/como-usar-el-comando-systemd-journalctl-para-administrar-registros/. [Último acceso: 2 07 2021].

[26] «bugs.launchpad.,» 18 05 2018. [En línea]. Disponible: https://bugs.launchpad.net/openstack-ansible/+bug/1771987. [Último acceso: 2021 07 21].

[27] Openstack, «Openstack,» 2021. [En línea]. Disponible: https://www.openstack.org/software/project-navigator/openstack-components#openstack-services. [Último acceso: 26 07 2021].

[28] Open Source MANO, «9. ANNEX 1: Troubleshooting,» 2020. [En línea]. Disponible: https://osm.etsi.org/docs/user-guide/09-troubleshooting.html. [Último acceso: 13 06 2021].

[29] «Conpilar,» 3 2021. [En línea]. Disponible: https://conpilar.es/como-usar-el-comando-systemd-journalctl-para-administrar-registros/. [Último acceso: 11 07 2021].

[30] «Ichi.pro,» [En línea]. Disponible: https://ichi.pro/es/ansible-una-de-las-mayores-bendiciones-en-esta-era-de-automatizacion-113523693217364. [Último acceso: 17 6 2021].

Anexo A. Archivo nat-net.xml

A continuación, se presenta el archivo de configuración de red que empleará KVM para la comunicación de las máquinas. En este caso se opta por una conexión NAT, la cual permite que las máquinas se comuniquen entre ellas y en adición, les sirve conexión a internet. En el apartado de configuración de red de host de despliegue viene explicado cómo hacer uso de este archivo y en qué comando incluirlo.

```
<network>

<name>NAT_mgmt</name>

<uuid>0935f331-7570-4ff8-a7b2-ee37ffe33774</uuid>

<forward mode='nat'/>

<bridge name='virbr_vm' stp='on' delay='0'/>

<mac address='52:54:00:4a:cc:eb'/>

<ip address='10.0.0.1' netmask='255.255.255.0'>

<dhcp>

<range start='10.0.0.2' end='10.0.0.254'/>

</dhcp>

</ip>

</network>
```

Anexo B. Configuración de red de los host

Cada host desplegado tiene una serie de necesidades en cuanto a red. En este anexo veremos la configuración de los archivos necesarios para desplegar las interfaces necesarias en cada una de las máquinas.

Host de despliegue

Archivo: /etc/network/interfaces

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Include files from /etc/network/interfaces.d:

source-directory /etc/network/interfaces.d

# interfaces(5) file used by ifup(8) and ifdown(8)

auto lo

iface lo inet loopback


# Container/Host management VLAN interface

auto virbr_vm.10

iface virbr_vm.10 inet manual

    vlan-raw-device virbr_vm


# Container/Host management bridge

auto br-mgmt

iface br-mgmt inet static

    bridge_stp off

    bridge_waitport 0

    bridge_fd 0
```

```
    bridge_ports virbr_vm.10

    address 172.29.236.16

    netmask 255.255.252.0

    #gateway 172.29.236.1

    dns-nameservers 8.8.8.8
```

Host Controller

Archivo: /etc/network/interfaces

```
# This is a single-NIC configuration to implement the required
bridges

# for Openstack-Ansible. This illustrates the configuration of
the first

# Infrastructure host and the IP addresses assigned should be
adapted

# for implementation on the other hosts.

#

# After implementing this configuration, the host will need to be

# rebooted.


# Physical interface

auto enp1s0

iface enp1s0 inet manual



# Container/Host management VLAN interface

auto enp1s0.10
```

```
iface enp1s0.10 inet manual
    vlan-raw-device enp1s0

# Openstack Networking VXLAN (tunnel/overlay) VLAN interface
auto enp1s0.30
iface enp1s0.30 inet manual
    vlan-raw-device enp1s0

# Storage network VLAN interface (optional)
auto enp1s0.20
iface enp1s0.20 inet manual
    vlan-raw-device enp1s0

# Container/Host management bridge
auto br-mgmt
iface br-mgmt inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports enp1s0.10
    address 172.29.236.11
    netmask 255.255.252.0
    dns-nameservers 8.8.8.8

# Bind the External VIP
auto br-mgmt:0
```

```
iface br-mgmt:0 inet static
    address 172.29.236.10
    netmask 255.255.252.0

# Openstack Networking VXLAN (tunnel/overlay) bridge
#
# The COMPUTE, NETWORK and INFRA nodes must have an IP address
# on this bridge.
#
auto br-vxlan
iface br-vxlan inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports enp1s0.30
    address 172.29.240.11
    netmask 255.255.252.0

# Openstack Networking VLAN bridge
#auto br-vlan
#iface br-vlan inet manual
#    bridge_stp off
#    bridge_waitport 0
#    bridge_fd 0
#    bridge_ports enp1s0
```

```
# compute1 Network VLAN bridge
auto br-vlan
iface br-vlan inet manual
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
#
#
auto br-storage
iface br-storage inet manual
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports enp1s0.20

# compute1 Storage bridge
#auto br-storage
#iface br-storage inet static
#    bridge_stp off
#    bridge_waitport 0
#    bridge_fd 0
#    bridge_ports eth0.20
#    address 172.29.244.12
#    netmask 255.255.252.0
source /etc/network/interfaces.d/*.cfg
```

Host Compute:

Archivo: /etc/network/interfaces

```
# This is a single-NIC configuration to implement the required
bridges
# for OpenStack-Ansible. This illustrates the configuration of
the first
# Infrastructure host and the IP addresses assigned should be
adapted
# for implementation on the other hosts.
#
# After implementing this configuration, the host will need to be
# rebooted.

# Physical interface
auto enp1s0
iface enp1s0 inet manual

# Container/Host management VLAN interface
auto enp1s0.10
iface enp1s0.10 inet manual
    vlan-raw-device enp1s0

# OpenStack Networking VXLAN (tunnel/overlay) VLAN interface
auto enp1s0.30
iface enp1s0.30 inet manual
    vlan-raw-device enp1s0
```

```
# Storage network VLAN interface (optional)
auto enp1s0.20
iface enp1s0.20 inet manual
    vlan-raw-device enp1s0

# Container/Host management bridge
auto br-mgmt
iface br-mgmt inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports enp1s0.10
    address 172.29.236.12
    netmask 255.255.252.0
    dns-nameservers 8.8.8.8 8.8.4.4

# OpenStack Networking VXLAN (tunnel/overlay) bridge
#
# The COMPUTE, NETWORK and INFRA nodes must have an IP address
# on this bridge.
#
auto br-vxlan
iface br-vxlan inet static
    bridge_stp off
    bridge_waitport 0
```

```
    bridge_fd 0
    bridge_ports enp1s0.30
    address 172.29.240.12
    netmask 255.255.252.0


# compute1 Network VLAN bridge
auto br-vlan
iface br-vlan inet manual
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
#
# compute1 Storage bridge
auto br-storage
iface br-storage inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports enp1s0.20
    address 172.29.244.12
    netmask 255.255.252.0
```

Host Block Storage

Archivo: /etc/network/interfaces

```
# This is a single-NIC configuration to implement the required
bridges
# for Openstack-Ansible. This illustrates the configuration of
the first
# Infrastructure host and the IP addresses assigned should be
adapted
# for implementation on the other hosts.
#
# After implementing this configuration, the host will need to be
# rebooted.

# Physical interface
auto enp1s0
iface enp1s0 inet manual

# Container/Host management VLAN interface
auto enp1s0.10
iface enp1s0.10 inet manual
    vlan-raw-device enp1s0

# Openstack Networking VXLAN (tunnel/overlay) VLAN interface
auto enp1s0.30
iface enp1s0.30 inet manual
    vlan-raw-device enp1s0

# Storage network VLAN interface (optional)
auto enp1s0.20
```

```
iface enp1s0.20 inet manual
    vlan-raw-device enp1s0

# Container/Host management bridge
auto br-mgmt
iface br-mgmt inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports enp1s0.10
    address 172.29.236.13
    netmask 255.255.252.0
    dns-nameservers 8.8.8.8 8.8.4.4

# Openstack Networking VXLAN (tunnel/overlay) bridge

# compute1 Network VLAN bridge
auto br-vlan
iface br-vlan inet manual
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
#

# compute1 Storage bridge
auto br-storage
```

```
iface br-storage inet static

    bridge_stp off

    bridge_waitport 0

    bridge_fd 0

    bridge_ports enp1s0.20

    address 172.29.244.13

    #gateway 172.29.244.1

    netmask 255.255.252.0
```

Host OSM

Archivo: /etc/network/interfaces

```
# This is a single-NIC configuration to implement the required bridges

# for OpenStack-Ansible. This illustrates the configuration of the first

# Infrastructure host and the IP addresses assigned should be adapted

# for implementation on the other hosts.

#

# After implementing this configuration, the host will need to be

# rebooted.


# Physical interface


source /etc/network/interfaces.d/*.cfg


auto lo
```

```
iface lo inet loopback

auto enp1s0
iface enp1s0 inet manual

# Container/Host management VLAN interface
auto enp1s0.10
iface enp1s0.10 inet manual
    vlan-raw-device enp1s0

# OpenStack Networking VXLAN (tunnel/overlay) VLAN interface
auto enp1s0.30
iface enp1s0.30 inet manual
    vlan-raw-device enp1s0

# Storage network VLAN interface (optional)
auto enp1s0.20
iface enp1s0.20 inet manual
    vlan-raw-device enp1s0

# Container/Host management bridge
auto br-mgmt
iface br-mgmt inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
```

```
    bridge_ports enp1s0.10
    address 172.29.236.15
    netmask 255.255.252.0
    dns-nameservers 8.8.8.8


# OpenStack Networking VXLAN (tunnel/overlay) bridge
#
# The COMPUTE, NETWORK and INFRA nodes must have an IP address
# on this bridge.
#
auto br-vxlan
iface br-vxlan inet static
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
    bridge_ports enp1s0.30
    address 172.29.240.15
    #gateway 172.29.244.1
    netmask 255.255.252.0

auto br-storage
iface br-storage inet manual
    bridge_stp off
    bridge_waitport 0
    bridge_fd 0
```

```
bridge_ports enp1s0.20
```

Anexo C. Archivo openstack_user_config.yml

El archivo openstack_user_config.yml es que usará ansible para saber qué servicios debe desplegar y sobre todo, en qué máquinas debe hacerlo. Las máquinas se indicarán por medio de su dirección IP. También en este archivo se declaran que rangos de IP existen para el despliegue y que máquinas se pueden conectar a cada red.

```
---

cidr_networks:

  container: 172.29.236.0/22

  tunnel: 172.29.240.0/22

  storage: 172.29.244.0/22


used_ips:

  - "172.29.236.1,172.29.236.50"

  - "172.29.240.1,172.29.240.50"

  - "172.29.244.1,172.29.244.50"

  - "172.29.248.1,172.29.248.50"


global_overrides:

  # The internal and external VIP should be different IPs, however
they

  # do not need to be on separate networks.

  external_lb_vip_address: 172.29.236.10

  internal_lb_vip_address: 172.29.236.11

  tunnel_bridge: "br-vxlan"

  management_bridge: "br-mgmt"
```

```
provider_networks:
  - network:
      container_bridge: "br-mgmt"
      container_type: "veth"
      container_interface: "eth1"
      ip_from_q: "container"
      type: "raw"
      group_binds:
        - all_containers
        - hosts
      is_container_address: true
      is_ssh_address: true
  - network:
      container_bridge: "br-vxlan"
      container_type: "veth"
      container_interface: "eth10"
      ip_from_q: "tunnel"
      type: "vxlan"
      range: "1:1000"
      net_name: "vxlan"
      group_binds:
        - neutron_linuxbridge_agent
  - network:
      container_bridge: "br-vlan"
      container_type: "veth"
      container_interface: "eth12"
```

```
    host_bind_override: "eth12"
    type: "flat"
    net_name: "flat"
    group_binds:
      - neutron_linuxbridge_agent
- network:
    container_bridge: "br-vlan"
    container_type: "veth"
    container_interface: "eth11"
    type: "vlan"
    range: "101:200,301:400"
    net_name: "vlan"
    group_binds:
      - neutron_linuxbridge_agent
- network:
    container_bridge: "br-storage"
    container_type: "veth"
    container_interface: "eth2"
    ip_from_q: "storage"
    type: "raw"
    group_binds:
      - glance_api
      - cinder_api
      - cinder_volume
      - nova_compute
```

```
###
### Infrastructure
###

# galera, memcache, rabbitmq, utility
shared-infra_hosts:
  infra1:
    ip: 172.29.236.11

# repository (apt cache, python packages, etc)
repo-infra_hosts:
  infra1:
    ip: 172.29.236.11

# load balancer
haproxy_hosts:
  infra1:
    ip: 172.29.236.11

###
### Openstack
###

# keystone
identity_hosts:
  infra1:
```

```
    ip: 172.29.236.11

# cinder api services
storage-infra_hosts:
  infra1:
    ip: 172.29.236.11

# glance
image_hosts:
  infra1:
    ip: 172.29.236.11

# nova api, conductor, etc services
compute-infra_hosts:
  infra1:
    ip: 172.29.236.11

# heat
orchestration_hosts:
  infra1:
    ip: 172.29.236.11

# horizon
dashboard_hosts:
  infra1:
    ip: 172.29.236.11
```

```
# neutron server, agents (L3, etc)
network_hosts:
  infra1:
    ip: 172.29.236.11

# nova hypervisors
compute_hosts:
  compute1:
    ip: 172.29.236.12

# cinder storage host (LVM-backed)
storage_hosts:
  storage1:
    ip: 172.29.236.13
    container_vars:
      cinder_backends:
        limit_container_types: cinder_volume
        lvm:
          volume_group: cinder-volumes
          volume_driver:
cinder.volume.drivers.lvm.LVMVolumeDriver
          volume_backend_name: LVM_iSCSI
          iscsi_ip_address: "172.29.244.13"

# placement
```

```
placement-infra_hosts:
  infra1:
    ip: 172.29.236.11
```

Anexo D. Error servicio Placement

En este anexo se recogen las trazas que se obtuvieron en el error mencionado en el apartado del despliegue con Ansible, así como la URL de la aportación con la solución a un foro de usuarios y desarrolladores.

```
TASK [os_nova : Run nova-status upgrade check to validate a
healthy                configuration]                **********
FAILED - RETRYING: Run nova-status upgrade check to validate a
healthy       configuration       (8       retries       left).
FAILED - RETRYING: Run nova-status upgrade check to validate a
healthy       configuration       (7       retries       left).
FAILED - RETRYING: Run nova-status upgrade check to validate a
healthy       configuration       (6       retries       left).
FAILED - RETRYING: Run nova-status upgrade check to validate a
healthy       configuration       (5       retries       left).
FAILED - RETRYING: Run nova-status upgrade check to validate a
healthy       configuration       (4       retries       left).
FAILED - RETRYING: Run nova-status upgrade check to validate a
healthy       configuration       (3       retries       left).
FAILED - RETRYING: Run nova-status upgrade check to validate a
healthy       configuration       (2       retries       left).
FAILED - RETRYING: Run nova-status upgrade check to validate a
healthy       configuration       (1       retries       left).
fatal: [compute1 -> 172.29.239.148]: FAILED! => {"attempts": 8,
"changed": false, "cmd": ["/openstack/venvs/nova-22.1.3/bin/nova-
status", "upgrade", "check"], "delta": "0:00:03.028331", "end":
"2021-06-23 10:08:48.680178", "failed_when_result": true, "msg":
"non-zero   return   code",   "rc":   2,   "start":   "2021-06-23
10:08:45.651847", "stderr": "", "stderr_lines": [], "stdout": "+-
------------------------------------------------------------
---+\n|                        Upgrade                        Check
Results                                                |\n+-------
------------------------------------------------------------
+\n|
Check: Cells v2
```

|\n| Result:
Success |\n|
Details: No host mappings or compute nodes were found. Remember
to |\n| run command 'nova-manage cell_v2 discover_hosts' when
new |\n| compute hosts are
deployed. |\n+-------------
----------+\n|
Check: Placement
API |\n| Result:
Failure |\n|
Details: Placement API endpoint not
found. |\n+-----------------------------
-----------+\n| Check: Ironic Flavor
Migration |\n| Result:
Success |\n|
Details:
None |\n+--

--+\n| Check: Cinder
API |\n| Result:
Success |\n|
Details:
None |\n+--

--+\n| Check: Policy Scope-based
Defaults |\n| Result:
Success |\n|
Details:
None |\n+--

--+\n| Check: Policy File JSON to YAML
Migration |\n| Result:
Success |\n|
Details:
None |\n+--

--+\n| Check: Older than N-1

```
computes                                                |\n| Result:
Success                                                   |\n|
Details:
None                                                    |\n+--
-------------------------------------------------------------
--+", "stdout_lines": ["+-------------------------------------
----------------------------+",      "|      Upgrade      Check
Results                                             |", "+-----
-------------------------------------------------------------
+",                                                        "|
Check: Cells v2                                           |",
"|                                                      Result:
Success                                                   |", "|
Details: No host mappings or compute nodes were found. Remember
to |", "|   run command 'nova-manage cell_v2 discover_hosts' when
new                    |",    "|         compute    hosts    are
deployed.                                          |", "+-----------
-----------------------------------------------------+",   "|
Check: Placement API                                      |",
"|                                                      Result:
Failure                                                   |", "|
Details:        Placement        API        endpoint        not
found.                          |", "+--------------------------
----------------------------------------+",  "|  Check:  Ironic
Flavor Migration                                    |", "| Result:
Success                                                   |", "|
Details: None                                             |",
"+-------------------------------------------------------------
------+",             "|             Check:             Cinder
API                                                 |", "| Result:
Success                                                   |", "|
Details: None                                             |",
"+-------------------------------------------------------------
------+",       "|       Check:       Policy       Scope-based
Defaults                                           |",  "| Result:
Success                                                   |", "|
Details: None                                             |",
```

```
"+------------------------------------------------------------
------+",    "|    Check:    Policy    File    JSON    to    YAML
Migration                                          |",  "|  Result:
Success                                                    |", "|
Details: None                                                |",
"+------------------------------------------------------------
------+",       "|       Check:       Older       than       N-1
computes                                          |", "| Result:
Success                                                    |", "|
Details: None                                                |",
"+------------------------------------------------------------
------+"]}

NO                  MORE                  HOSTS                  LEFT
*************************************************************
**********

PLAY                                                              RECAP
*************************************************************
******************
compute1                                                          :
ok=97   changed=38   unreachable=0    failed=1    skipped=42   r
escued=0                              ignored=0
infra1_cinder_api_container-31aa2360                              :
ok=154  changed=73   unreachable=0    failed=0    skipped=30   r
escued=0                              ignored=0
infra1_glance_container-571d64f9                                  :
ok=123  changed=64   unreachable=0    failed=0    skipped=22   r
escued=0                              ignored=0
infra1_keystone_container-ab344f56                                :
ok=127  changed=61   unreachable=0    failed=0    skipped=42   r
escued=0                              ignored=0
infra1_nova_api_container-a8fa911b                                :
ok=146  changed=72   unreachable=0    failed=0    skipped=32   r
escued=0                              ignored=0
localhost                                                         :
ok=4    changed=3    unreachable=0    failed=0    skipped=0    r
```

```
escued=0                                    ignored=0
storage1                                                                  :
ok=90    changed=46    unreachable=0    failed=0    skipped=40    r
escued=0                                    ignored=0



EXIT       NOTICE       [Playbook       execution       failure]
************************************
=============================================================
===============
root@mike:/opt/openstack-ansible/playbooks#
```

URL con la solución aplicada: https://bugs.launchpad.net/nova/+bug/1671166/comments/8 . [Último acceso: 24 07 2021].

Anexo E. Contenido archivo admin-openrc.sh

En este anexo se muestra el contenido del archivo que se debe ejecutar en cualquier máquina de la infraestructura Openstack antes de poder usarla. Dicho de otra manera, con este archivo se le incluyen a la máquina las variables de entorno necesarias para establecer comunicación con los servicios.

```
#!/usr/bin/env bash

# To use an Openstack cloud you need to authenticate against the
Identity

# service named keystone, which returns a **Token** and **Service
Catalog**.

# The catalog contains the endpoints for all services the
user/tenant has

# access to - such as Compute, Image Service, Identity, Object
Storage, Block

# Storage, and Networking (code-named nova, glance, keystone,
swift,

# cinder, and neutron).

#

# *NOTE*: Using the 3 *Identity API* does not necessarily mean
any other

# Openstack API is version 3. For example, your cloud provider
may implement

# Image API v1.1, Block Storage API v2, and Compute API v2.0.
OS_AUTH_URL is

# only for the Identity API served through keystone.

export OS_AUTH_URL=https://172.29.236.10:5000

# With the addition of Keystone we have standardized on the term
**project**
```

```
# as the entity that owns the resources.

export OS_PROJECT_ID=bac5c9ec550e43c69e6a1f2dc00c349b

export OS_PROJECT_NAME="admin"

export OS_USER_DOMAIN_NAME="Default"

if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset OS_USER_DOMAIN_NAME;
fi

export OS_PROJECT_DOMAIN_ID="default"

if [ -z "$OS_PROJECT_DOMAIN_ID" ]; then unset
OS_PROJECT_DOMAIN_ID; fi

# unset v2.0 items in case set

unset OS_TENANT_ID

unset OS_TENANT_NAME

# In addition to the owning entity (tenant), Openstack stores the
entity

# performing the action as the **user**.

export OS_USERNAME="admin"

# With Keystone you pass the keystone password.

echo "Please enter your Openstack Password for project
$OS_PROJECT_NAME as user $OS_USERNAME: "

read -sr OS_PASSWORD_INPUT

export OS_PASSWORD=$OS_PASSWORD_INPUT

# If your configuration has multiple regions, we set that
information here.

# OS_REGION_NAME is optional and only valid in certain
environments.

export OS_REGION_NAME="RegionOne"

# Don't leave a blank variable, unset it if it was empty

if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
```

```
export OS_INTERFACE=public

export OS_IDENTITY_API_VERSION=3
```

Anexo F. Error de integración entre OSM y Openstack

En este anexo se recogen las trazas que se obtuvieron en el error mencionado en el apartado de la integración entre OSM y Openstack, así como la URL de la aportación con la solución a un foro de usuarios y desarrolladores.

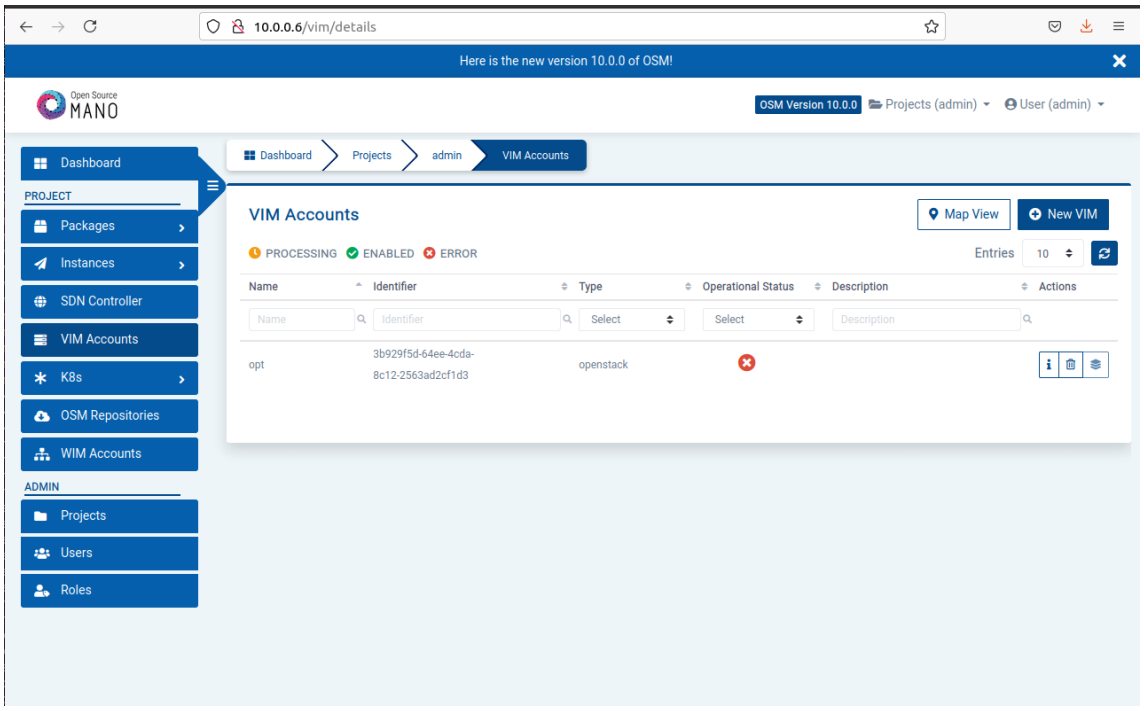

Ilustración 34. Error de integración entre OSM y Openstack.

URL a la resolución del problema: https://stackoverflow.com/questions/66853372/encountered-problem-while-integrating-devstack-osm-open-source-mano/68398891#68398891

Anexo G. Repositorio GitHub

De cara a futuros despliegues, se ha habilitado un repositorio Git para clonar y agilizar el abastecimiento de los principales archivos.

- https://github.com/MigueelHeerrero/OpenStack.git

Anexo H. Archivo de credenciales

```
adjutant_galera_password:
70c82f6a9da9f52374d5bac2d20c269ec9028373b37654fd8bb071e063

adjutant_secret_key: d233e2e087997fd09b30b76484519aec

adjutant_service_password: 1be1fe65f6bb10d149da2ee38e

aodh_container_db_password:
0f8abd000222b88dd52cabc9803d5b2c0ccfe5bf01d4e2df047fe63d

aodh_oslomsg_rpc_password:
0f9568f57f13688cef10b9a0a1bda057e94d84f9eb32a9e0ff1

aodh_service_password: 273ec75de0b912b48f1dff11b8228b

barbican_galera_password: 5b053ccc83e11ef95add645e7e77e534b6798e

barbican_oslomsg_rpc_password:
c26bd00e83d44d3b57944bc6c16423226f98ec1284

barbican_service_password:
21b3907a59b1b79e005590d6c2443e63f486f283b53e01cd4d1af93

barbican_simple_crypto_key: b8fb4506cdce55be7272ebd456ed3773

blazar_container_mysql_password:
fb0c06418aaae59772b3abafa6e1bd1765a

blazar_oslomsg_rpc_password:
e9ae2614cca7222ed5cf9d06e9e03b68e27053858679

blazar_service_password: 38987bc515236b35db

ceilometer_container_db_password:
03acbb1941e75e31c76e79f701797ffc8df192c2d

ceilometer_oslomsg_rpc_password:
545003b8b4ab98f8fadada8ceb77f6e5c130

ceilometer_service_password: fefd4119a0cd2d1ceef563287b663
```

ceilometer_telemetry_secret:
2a0d505016dd32bc83e42aec2e31927462da5aa99c8828e6835780

cinder_ceph_client_uuid: de01aa88-8f88-41d3-b87d-258325281969

cinder_container_mysql_password:
996068a5c12e9b8ee30a7f994820f20a1933445816f453e39207ed395281c9

cinder_oslomsg_rpc_password: e32043be57df32b082a6ce6d8e859fdfcad

cinder_profiler_hmac_key: 93911df902a764b1fa2ef5d376c3b36a

cinder_service_password: 91adb65d100aef49749

cloudkitty_container_mysql_password: 6948ae7c7626326523936edc0b5

cloudkitty_oslomsg_rpc_password:
301e049ee099ee09c87b7163d23acbaf3f8e5bb5839abe50ad3

cloudkitty_service_password:
dbfa38210eb68f6714d3083f7044496f1760b5faf67f9adaa2

designate_galera_password:
7f62f08831715bc9fd73e6ab8fa878ac2645d5

designate_oslomsg_rpc_password:
6d46024e8adfbe18c77a3e0b199f0fc34882ae285baf45

designate_pool_uuid: acf77ca1-8c9f-4d4e-b45c-d90dbf060246

designate_service_password: 0dc4aeb75625fb458641f5e

galera_mariadb_backups_password:
b7a5925ed1083154181b2adc445d7a334b2a63de47e208abddccc866761

galera_root_password: fbc463a7bc117067eeb8e812463798

glance_container_mysql_password:
9e62b63bbe1cc66806ea47d9045c9d60d996084c2fc99171eb6cb81

glance_oslomsg_rpc_password:
fe00b9631f58b5ccb73183b4ec8341ea6d61f721435110

glance_profiler_hmac_key: d46e7e159fb90a46f32ccc53aaff3efd

glance_service_password:
40cd58867c186a2f390ebc118a10ebce47b56ba5dcd1635ff

gnocchi_container_mysql_password: 03053b6a62708181e634454

gnocchi_service_password: d9d82109d991e256e7ff7d93e2d4d71b

haproxy_keepalived_authentication_password:
6e322c31008d858f1b221e122d1f860632f7c2c644d073

haproxy_stats_password: 974aa8a80f2d1c7b1b04108ea47994

heat_auth_encryption_key: 21b8b6e8ef95e831db292c5acc4f726e

heat_container_mysql_password: b9002de0e560e32328e8ffa2627a1d

heat_oslomsg_rpc_password: 688e00045c4819bb8afe06e8e368c087c628a

heat_service_password: fedc000af6b330ca4d40ad2

heat_stack_domain_admin_password:
7d927156ef7b1cc09549ce31988149b3fcab85a1

horizon_container_mysql_password:
7980da274ba2ed8fed0985f75c51d453c67e93e561ebc4

horizon_secret_key: 1fd577269e5e9ecb55c8a18b225524b5

ironic_container_mysql_password:
511c02b5fab57c97c57622ef5d06f70b9b69a53e8c05f86df749b1b6fabbc1

ironic_inspector_container_mysql_password:
045f9a78273ea5aab4050a9898a08e95d394da6926dc9bfe4dab8a0

ironic_inspector_service_password: c3dd6b3d46f4a1394ec

ironic_inspector_swift_password:
9051b5de84065ada75ee7203282bc4287d914ad957e6

ironic_oslomsg_rpc_password:
fc79e49de9dc01b4329bad595f9945cce6052f1758c17132b

ironic_service_password:
3ef4b2096a27fa04fbbe88444fc53c8437bd538d0d

ironic_swift_temp_url_secret_key:
a553dcc8b2f07f1331388179bcf2e093

keystone_auth_admin_password: 7d952e2c37d6d586edeebb7cd4487935e

keystone_container_mysql_password:
8408f298a7e3227e66314d624b0ca86c5cb786dda6edb06ef3ee5

keystone_oslomsg_rpc_password:
3bf2f720aeba311125914660e63b9f17f1ce29cdaf1716ca46f

magnum_galera_password:
8c6424c939a828deaae2969a58b0b6cb1257750fc00283ffb21f7d2

magnum_oslomsg_rpc_password:
224ae9bae958ad2dca63a1d435d6b6a27952

magnum_service_password: 944e547ac35f12aebe3e418cda

magnum_trustee_password:
16f907c880d09bc67b82ceeb50e9ed9d5bd23652ff0edb8c2f7253bf20

manila_container_mysql_password:
3e7d5f03482e9be9542dc58c06c8f5e010a6add8545526

manila_oslomsg_rpc_password: 5d7f3b72b3ed0486699e

manila_profiler_hmac_key: 34202c447ddfce94b0eca79eca926e2e

manila_service_password: 7a17bf2d3f77f598826a3bb62a73c5cf95

masakari_container_mysql_password:
4bc58ef6bd45843b76ae69dccc5f624bb68098a7f274df

masakari_oslomsg_rpc_password:
74147a22b5343fc1d9a56ac4466439dc40a4a3d6fc36f359175b98831c15

masakari_service_password:
134ff1cc521fa93fb994dbe49d5427316f390ba244b4e151436847a9c

memcached_encryption_key: ca79b2b64b17f05886fcf9ded23b13e3

mistral_galera_password:
54bde58ef680b11879011afd42371e8543ae945c73492ea08d

mistral_oslomsg_rpc_password: 496b86b95eaf98abdef10

mistral_service_password: 2ab9f56cbe28fd866

murano_galera_password: c7221acde6aa0f856464fdd

murano_oslomsg_rpc_password:
a1f71721a0e9e591f4227832bcec7cf86b62a5c5a3122dab54105f7a0d

murano_service_password: c9a709f57bf997528

neutron_container_mysql_password:
e950d2019557cff9103b4c617153f4c420f9a4092e2d0f

neutron_ha_vrrp_auth_password:
c7bfdd34a6d1fd17bf94e1606633a1714deb379db460190e870

neutron_oslomsg_rpc_password:
70cfde6137ce80c487a33d9485d4a3ef6eb3b26

neutron_service_password: 753f8dbba3cb88ad3cc7e5aaa9ed7e91006c8

nova_api_container_mysql_password:
6be6a1cba3dffa094899812e5981179486c876e78867

nova_container_mysql_password:
6072ff28c45a037a1c10df646c42ec47c1b2946e76df79453ff710ec73

nova_metadata_proxy_secret:
0656ddaac5a86b1c00379374d4e1176a4ea12139121581e17b3e36

nova_oslomsg_rpc_password:
883a4d222557b398225e8f5af5bd623ff68cc73f6d3529ea4f1

nova_service_password:
947323f89eec92a5612bcfae7cd225100a5953a207

octavia_cert_client_password:
308536be29182320a3c8405715aad2e9be2

octavia_container_mysql_password: 4c1a0c49c0085f0c5

octavia_health_hmac_key: 38f498f3ca649aa8f77289b355853961

octavia_oslomsg_rpc_password:
a634bba3e3fc18352764a701167d180dda0e7fc324a58fd4406

octavia_service_password: 15142ca8612ba56c969898

panko_container_db_password: ae966830f665c25109776c

panko_service_password:
dc97e18f41e086bc43098d95d55b41f3dabffc020cf59dae7b5f

placement_galera_password:
35e0469117b422893730b7fedafc07cc8a1f5ac62e782682ff

placement_service_password:
09735907a3350ffe18a1e2c29950c223ba5a7

rabbitmq_cookie_token:
a2d1e3c315aaaa127d687768fc251d3ff8029106e4f220cd4cf5aa7c197269568d

rabbitmq_monitoring_password: 1cf9e0e3e91940e8d

radosgw_admin_password:
8d9b2e0d29ad3d047657287842f343c9289d4b332ac55b1

rally_galera_password: e1bd877e68e20bac8e65ddfb

sahara_container_mysql_password:
f88f99b5d79cb9bfc606e53cd75b6b25

sahara_oslomsg_rpc_password:
f76db84ffedebf70997095e31f9cf08c069133a3f358

sahara_service_password: fdb14494d642ce7c24640a5f63f7d2b140

senlin_galera_password:
71638aaa2428ed4593f723ea90b484a32ea67f372e1da79d

senlin_oslomsg_rpc_password:
0516b2e74462493567f806dd30b4d506497f68

senlin_service_password: cfd28c859dda429432a7b2940

swift_dispersion_password: 90975cc23f7edcbd

swift_hash_path_prefix: b806e568351abbb4abcd02605411e263

swift_hash_path_suffix: c760d140f6b2f185be855501f1727f50

swift_oslomsg_notify_password:
b184b7f1fe75a699ea87bca83acec180e3d9

swift_service_password: 6ce95fc2e8a576c2795fc2d249a7a285a8ba44b

tacker_container_mysql_password:
bb3fac3c3709f978041ab1fce931661ed9480a390510382a5162d

tacker_oslomsg_rpc_password: 7497c7ba0dfd1a67c8

tacker_service_password: 9f8a156376747025

trove_admin_user_password:
d6a662702157046a571a669ffa9cc7fd49580c1893cd8

trove_galera_password:
e3423ccf53ec13aa7bf60050d7f545a94ec064606be

trove_inst_rpc_key_encr_key: ff7c3f192913fc56cf0312563424459d

trove_instance_rpc_encr_key: 98e33278b1d07caee438c0d899a03b37

trove_oslomsg_rpc_password: 00cf5d03ba5a335d48c73706e81

trove_service_password: ba4bd94a3ba109662bbf779bca044c616c8

trove_taskmanager_rpc_encr_key: af30b19c8e636a59e828569707fc3118

zun_galera_password:
0499c2edbf9e164097187dfd169361878d666ca0b2b2b2600983f68

zun_kuryr_service_password:
837f2247a4b9778b1b9e8659481137b2afd32aaebf17a2585754d642cc809b

zun_oslomsg_rpc_password:
6b2f1ac0b364cc33427c68bfa5a7da3e582e54323314513f737e939d3500

zun_service_password:
5e17b074a983b7bad10ce026cee4f4c534b831ca7b6cc173a9

Universidad de Alcalá